Title

Data Driven Algorithms For Perception With Applications To Autonomous Driving, Energy And Mixed Reality

Author

Afolabi, Oladapo

Publication Date

2020

Peer reviewed|Thesis/dissertation

Data Driven Algorithms For Perception With Applications To Autonomous Driving, Energy And Mixed Reality

by

Oladapo Afolabi

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor S. Shankar Sastry, Chair
Professor Yi Ma
Professor Luisa Caldas

Summer 2020

Data Driven Algorithms For Perception
With Applications To Autonomous Driving, Energy And Mixed Reality

Abstract

Data Driven Algorithms For Perception
With Applications To Autonomous Driving, Energy And Mixed Reality

by

Oladapo Afolabi

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor S. Shankar Sastry, Chair

The rise of autonomous and artificially intelligent systems promises to deliver efficient and optimal performance as well as unprecedented functionality in a variety of fields and human endeavors. As varied as these fields are, a common property of most autonomous systems is that they convert raw data about their relationship with the environment into optimal actions to achieve a desired goal. One may further divide this conversion into the sub-tasks of sensing and perception, planning, and actuation. Perception is an important part of this pipeline since it describes the set of algorithms for turning raw data into useful bits of information that can be used for planning.

At its core, we may view perception using an estimation framework. Perception algorithms essentially seek to estimate information from noisy, incomplete and raw data obtained from sensors. However, the accuracy of the information obtained (and consequently effectiveness of the actions taken by the system) are dependent on the assumptions and models used in designing perception algorithms. Therefore, it is important to take care in designing accurate but tractable models for perception. Unfortunately, many perception tasks involve complex functions and sensor models relating observed data to information. These functions are difficult to model from physical properties. In these cases data-driven methods can provide complementary techniques that result in excellent models for very complex systems.

In this dissertation, we present three perception algorithms designed for applications in Autonomous Driving, Energy Systems and Mixed Reality. We make use of data driven methods to provide approximations to complex sensor models and present tractable estimation algorithms for turning raw data into information. We verify our approach on both synthetic and real data and report excellent results.

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

My journey to and through graduate school would not have been possible without the help of so many people who have encouraged me and given me the opportunity to conduct the research presented in this document. As a sign of gratitude, I would like to acknowledge some of the most significant contributors to my success. First, I would like to thank God for having the resources and opportunity to pursue my dreams at a top academic institution. I realize I have been quite fortunate to have a lot of things fall in place so that I can be where I am today. I would like to thank my parents for their unwavering support, their encouragement to keep pushing at times when I thought it would be better to quit. On many occasions, I have questioned the reasoning behind their insistence on completing this degree, but looking back I am glad I took their advice and faced my fears. I would also like to thank my sisters Tolani and Bolaji for helping me complete the best trio I could ever imagine. You have been both a source of encouragement and a rock to lean on. I thank my extended family and especially my cousins for their support throughout graduate school. Your phone calls have helped me get through some of my tougher times.

I am grateful to my advisor, Professor S. Shankar Sastry for his encouragement and patience. His taste for rigorous work will always be the standard I aspire to. I am grateful for his guidance with regards to how to find good research problems. I am also grateful to Dr. Allen Yang, who introduced me to the wonderful world of computer vision and has been the instigator of many research problems I have worked on. This thesis would look very different without his mentoring. I am grateful to Professor Yi Ma, for his inspiration and guidance on 3D vision, as well as his encouragement to put in the hard work. I would also like to thank Professor Luisa Caldas, for her excellent feedback on my work from a non-engineering perspective. I would like to thank Professor Roy Dong and Professor Katherine Driggs-Campbell for helping me out when I had run out of ideas. I truly think my graduate school career was saved through the collaborations we had together.

One of the best parts of graduate school has been getting to make friends. I have learned so much about life and research from my friends. I would like to thank Sarah Seko, Vicenç Rubies-Royo, Margaret Chapman, Carlos Baiou, Casey Mackin, Robert Matthew, Ben Osoba and Sylvia Herbert for the time spent together and conversations both in and out of class.

I would like to thank my research group as well as Claire Tomlin's group, Ruzena Bajcy's group and Mohammad Keshavarzi for the rich conversations we had. In addition, I thank all the undergraduate and master's students I have worked with. Your persistence and innovative spirit have motivated me to do more.

Finally, I would like to thank all those in the EECS community who have been very friendly and helpful. Your actions made me feel very welcome. I would like to especially thank Jessica Gamble and Shirley Salanio for their excellent work and Sheila Humphreys for always reaching out.

This work is supported in part by the Office of Naval Research (ONR) under grant N00014-19-1-2055 and also based upon work supported by the Office of Naval Research (ONR) under grant number 31701-23800-44–EHS1S, the NSF Graduate Research Fellowship

under grant DGE 1106400, NSF CPS:Large:ActionWebs award number 0931843, TRUST (Team for Research in Ubiquitous Secure Technology) which receives support from NSF (award number CCF-0424422), and FORCES (Foundations Of Resilient CybEr-physical Systems), the European Research Council under the advanced grant LEARN, contract 267381, a postdoctoral grant from the Sweden-America Foundation, donated by ASEA's Fellowship Fund, and by a postdoctoral grant from the Swedish Research Council.

Chapter 1

Introduction

1.1 What is perception and why perception?

"All our knowledge begins with sense, proceeds thence to understanding, and ends with reason"[51]. This statement is as true for humans as it is for artificially intelligent and autonomous systems.

Much like humans, autonomous systems are designed to observe the relationship between their environment and themselves, extract meaningful information from these observations, and act on this information to modify the relationship in ways that achieve a desired set of outcomes. From this standpoint, one may broadly decompose autonomous systems into three parts:

1. Sensing: This refers the process of collecting data about the relationship between an autonomous system and the environment, as well as processing this data to produce useful information. Data on the relationship between an autonomous system may comprise of observations about processes in the environment or the internal state of the autonomous system.

2. Planning: Planning refers to the process and set of algorithms that convert information to a set of actions required to achieve desired outcomes.

3. Actuation: Actuation refers to the process of either changing the system's internal state, the environment or both. For autonomous systems, actuation is the physical realization of the actionable outcomes from the planning stage.

For example, in mobile robots, the sensing may comprise of physical sensors such as cameras, LIDAR sensors, wheel encoders and algorithms to turn the raw sensor reading into meaningful information. Such algorithms may include Localization and Mapping algorithms or image classification algorithms. Planning may include algorithms to convert information such as the pose of the robot and location of key objects in the environment into a useful set of actions. In this scenario, this may simply be a path planning algorithm that outputs an

obstacle free path to the exit in a room. Actuation would involve moving the robot along this path towards the exit.

However, actuation need not always involve movement or mechanically moving parts. In our broad definition, actuation can be any response made by the autonomous system to effect a change on the environment or itself. For example, an energy disaggregation system estimating the power consumption of individual appliances from aggregate power measurements may display the resulting disaggregated information or a warning about appliance usage as its form of actuation.

Sensing and Perception

This breakdown is just one way to view autonomous systems. The observant reader may have noticed that we may further decompose sensing into data collection and data processing. In fact, one could argue that the data processing stage could be another component on its own or even part of the planning stage.

This argument may be valid. For simplicity we will stick to our original decomposition but emphasize the difference between sensing and perception. As stated above, sensing may be decomposed into two parts, data collection and data processing. It is this data processing component we refer to as *perception*. Analogous to how we may look around a room and observe large slabs of white boundaries leading to a brown rectangular pattern and perceive it as a set of walls leading to a door, or the set of contours and lines on a page and perceive it to be the drawing of a face, perception is what turns vast amounts of observed data to manageable and meaningful abstractions called information.

Without this information autonomous systems run into difficulty when trying to make sense of their environment, much in the same way a patient with Wernicke's aphasia may have trouble reading or recognizing a person even though they see the strokes making the words on a page and the lines contouring a face; hence the importance of perception. It is this definition of perception that is the focus of this thesis.

A Mathematical tool for perception

Another reason we will stick to our decomposition of autonomous systems is that the decomposition lends itself easily to powerful mathematical tools for implementing sensing modules. Concretely, estimation theory is an area of statistics concerned with estimating the values of a set of parameters based on observed data. Once we define the information we are after as a function of these parameters, it is easy to see why estimation theory should play an important role in perception algorithms. In addition, since in practice data observed is always impacted by some form of uncertainty e.g. sensor noise, occlusion etc., statistical frameworks seem to be the right set of tools of the job.

We will make use of Maximum A Posteriori (MAP) estimators, for which it is necessary to define the following:

1. Sample space and Data: This refers to the space of observations and set of observed data points that will inform our choice of optimal parameters. The data observed belongs to the space $\mathcal{Z}$ and is a realization of a random variable Z.

2. Parameter space, parameters and functions of parameters: This refers to the set of the parameters we wish to estimate, X, taking on values from some parameter space $\mathcal{X}$ or possibly functions of these parameters $h(X)$. These parameters and their image under the various functions of interest will correspond to the information we seek to obtain from sensing. X is a random variable.

3. Likelihood function and prior density: To model the uncertainty in our measurement and to incorporate partial knowledge about the set of parameters we wish to estimate, it is necessary to specify distributions of the data conditioned on the values parameters may take. This distribution is referred to as the likelihood function $p_{Z|X}$. One may also specify a distribution over the set of parameters if that knowledge is available. We refer to this as the prior distribution p_X.

To model the uncertainty in the observed data, we will model the process relating our data and parameters as a function of the parameters of interest with added noise. We refer to this model as an *observation model*. Concretely, we have that:

$$Z = h(X) + W \ ,$$

where the noise W is a random variable following some known distribution p_W and $h(.)$ is our *sensor model*, informing us of what type of observations we should expect given a realization of the random variable X. Characterizing the exact noise model from first principles can be difficult and it is common to make assumptions on the distribution such as it being Gaussian or Laplacian. The justifications for such assumptions typically appeal to the Central Limit Theorem as well as ease of optimization.

With this choice of an observation model, the likelihood function will fall into the same family of distribution as the noise variable. Our MAP estimator, which is a function that approximates the realized parameter value from the observed data i.e. converts collected data to information is:

$$\hat{X}(z) = \underset{x \in \mathcal{X}}{\operatorname{argmax}} \ p_{Z|X}(z|x) p_X(x) \ . \tag{1.1}$$

Simply put, our estimate of the true parameter (and information) is the one that maximizes the posterior probability. In certain cases, we may want more information than just the mode of the posterior distribution. In these cases we may instead compute the posterior distribution entirely. The benefit of this is that we can qualify the uncertainty around our estimate. In the algorithms presented in this thesis, we have made use of both alternatives, choosing to use the mode of the posterior when it is practical and computing the entire posterior when it is useful and feasible.

Why data driven?

This framework for extracting information from data is theoretically sound, however herein lies the problem. Our estimated parameters and hence obtained information are only as good as the models we use. Poor sensor models are bound to result in poor parameter estimates. As the saying goes "Garbage in, garbage out".

It is therefore crucial to obtain accurate models of whatever noise variables or sensor models we will be making use of. Unfortunately, modeling noise as well as sensors used in autonomous systems can be a grueling task. However, while we may be content with making certain assumptions about noise distributions, data driven models may be used to obtain sensor models approximating the behavior of autonomous systems.

Complex autonomous systems including autonomous driving, mapping and robotic grasping systems have benefited from data-driven models [92, 104, 74]. This is not to say that we should ignore physics based models when we are able to use them. When certain physical characteristics of the sensor are known, it is wise to make use of these characteristics in modeling and then restrict the data driven portion to those parts of the model we have little information about. We will take a this approach in the algorithms presented in this work incorporating both physics and data driven models whenever possible to improve our estimation performance.

Roadmap

The range of application of autonomous and artificially intelligent systems to the various facets of life is limited only by imagination. We see many examples ranging from individual scale application such as personal robotic systems and virtual reality headsets to large scale communal applications such as autonomous highways and energy grids. In this thesis we focus on three applications, namely Autonomous driving, Energy disaggregation and Mixed Reality systems. In each scenario, we will present data driven perception algorithms, their implementation and performance. In chapter 2, we present a method for pedestrian detection in autonomous driving scenarios that focus on cases where the physical sensors mounted on a vehicle might fail or be occluded. We make use of the data-driven behavioral models of other human drivers to predict possible pedestrian locations. In chapter 3, we present an energy disaggregation algorithm that estimates individual appliance power consumption from measured whole house energy consumption. We make use of data-driven appliance models as in this case modeling individual appliances from physical principles is not realistic. Finally in chapter 4, we present a a 3D reconstruction and perception algorithm geared towards mixed reality applications. We make use of data-driven 3D shape models to reconstruct 3D shapes and their poses from partial observation.

In each chapter, we have decided to maintain notation that is common in the literature. Unfortunately, this conflicts with the goal of sharing notation across the chapters. To improve the ease of understanding the material we will include a table of notation along with each

chapter. We end with discussions on the work presented in this thesis as well as exciting suggestions for future research directions.

Chapter 2

Autonomous Driving: People as Sensors

List of Symbols

$\mathcal{M}, \mathcal{A}$	Space of maps and actions respectively.
$X_{1:t}, Z_{1:t}, A_{1:t}$	Set of random variables representing pose, observed environmental data and actions from time 1 to t respectively.
x, z, a	Specific values of pose, observed data and action respectively.
m_i	value of i'th gridcell (used for occupancy grid representation of the environment).
ξ_i	value of i'th feature/landmark (used for landmark representation of the envronment).
M_i	Random variable representing the i'th grid cell.
Ξ_i	Random variable representing the i'th feature/landmark.
$\mathbf{M}$	Random variable representing the map of the environment ($\mathbf{M} = \{M_i\}_{i=1:n}$ for occupancy grid representation and $\mathbf{M} = \{\Xi_i\}_{i=1:k}$ for landmark representation).

2.1 Introduction

Our first example of a data-driven perception algorithm finds its application in autonomous driving. Despite growing attention to autonomous driving, there are still many open problems, including how autonomous vehicles will interact and communicate with human agents [32]. These concerns are particularly important when considering vulnerable users like pedestrians [18]. Although there has been some work in vehicle control in the presence of pedestrians,

the majority of research has been focused on improving perception for pedestrian detection [11, 38, 24].

While detection is important for a complete autonomous system, this chapter considers a specific scenario concerning the interaction between pedestrians and drivers, and examines how that interaction might influence map estimation, as a proxy for detection. Such a scenario is shown in Fig. 2.1. In this scene, a pedestrian may be starting to cross the street. From the perspective of the red car, the human is occluded. We examine how to take advantage of other agent's actions to infer the presence of a pedestrian despite occlusion. We present an approach based on an a-posteriori estimation framework.

Mapping in mobile robotics refers to the process of representing an agent's environment. Based on this representation of the environment, the agent can make intelligent decisions on how to behave in and interact with the environment. In our scenario, the agent is the ego (red) vehicle.

One common representation of the environment is the occupancy grid map. The occupancy grid map represents the environment as a grid of cells whose occupancy is modeled by independent binary random variables [34]. An occupancy grid map allows for tractability in representing large environments with considerable amount of detail and provides a starting point for more advanced representations. Another type of map representation is the sparse feature-based landmark map, which only represents key objects in the environment [41]. There are many more representations, many of which are more detailed (e.g. point-clouds and textured meshes), yet require more computational resources [105]. The particular choice of representation is dictated by the environment, computational resources, and how the map representation will be used to make decisions.

Common to all these representations is the need for sensor modeling. Sensor models are regularly derived from physical properties of how the sensor in question works. For example, the pinhole model is used for visual cameras and beam models for LIDAR and ultrasound sensors [106].

In this chapter, we exploit the fact that, aside from the physical interaction of energy (e.g. light and sound) with the environment, the actions of other intelligent agents also give us useful information about the environment. As such, we derive a data-driven behavioral model for agents in the environment and incorporate these people as sensors.

Behavioral driver modeling is an active area of research in many different applications, ranging from driver assistance systems to improved interaction for autonomy [32, 92, 28]. In [31], the mapping of the environment state was learned with respect to the states of the surrounding vehicles. These influences can also be learned by estimating the cost function of the driver, which can determine what actions the driver might take given some feature representations [2]. Driver models specifically considering pedestrian interactions have also been developed [88, 40, 99].

Few approaches have directly modeled the external influences of driver behavior in a manner that is amenable to improving environment estimation. Map-based approaches require directly modeling the connection between observable states of the vehicle (i.e., that which can be observed from a nearby vehicles) and the belief over the environment. In this

Figure 2.1: Motivating example of occluded pedestrian. *(Left)* Topview showing the red ego vehicle, the black car causing occlusion, and the hidden pedestrian. We model the actions of the black car as sensor inputs. *(Right)* Viewpoint from the red car (ego vehicle), showing that the pedestrian is occluded.

chapter, we focus on developing a driver model that can act as a sensor for the environment. We apply learning techniques to approximate the distribution over pre-determined driver behaviors given a map representation. From this, we integrate the sensor model into mapping frameworks to improve our overall awareness of the environment. This chapter presents four key contributions:

1. We introduce and formalize the concept of *people as sensors* for imputing maps.
2. We conduct an experiment with human drivers in a vehicle simulator to collect data on interactions between drivers and pedestrians.
3. We demonstrate improved environment estimation using occupancy grids on the collected data;
4. We modify pedestrian motion estimation and prediction in a landmark representation of mapping and test on a real-world dataset.

This chapter is organized as follows. The methodology used to integrate driver models and mapping is summarized in Section 2.2. The experimental setup for the user studies is described in Section 2.3. Section 3.8 presents our results using our dataset. Our method is also validated using an existing real-world dataset in Section 2.4. Section 2.6 discusses our findings and outlines future work.

2.2 Methods

This section provides a brief overview of mapping and the methods used to incorporate people as sensors.

Mapping Preliminaries

Let $\mathcal{M}$ represent the space of maps. A map $\mathbf{m} \in \mathcal{M}$ represents a possible state of the environment. In addition, let $x_{1:t}$ represent relevant information about the mobile agent up to time t (e.g., pose) and $z_{1:t}$ represent information about the physical state of the environment up to time t, (e.g., position of other vehicles in the environment). Further, we will interpret these quantities as realizations of the random variables $\mathbf{M}, X_{1:t}, Z_{1:t}$ respectively. Then, the problem of mapping can be formulated as estimating the posterior belief at time t, $p_{\mathbf{M}|X_{1:t},Z_{1:t}}(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t})$, over the space of maps $\mathcal{M}$. To simplify notation, we will drop the subscript on the distribution when it is clear from the context what distribution we are referring to.

The choice of environment representation largely determines which algorithm to use to estimate $p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t})$. In this chapter, we have chosen to represent the environment using two different approaches, depending on the structure of the data. We examine applying the people as sensors framework to

1. mapping the world using occupancy grids.

2. mapping the world using a collection of sparse landmarks in the environment.

Occupancy Grid Maps

When the environment is represented using an occupancy grid, the world is a set of binary random variables arranged in grids. Each random variable indicates whether or not its corresponding grid cell is occupied. Therefore, each map $\mathbf{m}$ is a realization of a set of binary random variables. If we denote the value of the grid cell with index i as m_i, (with corresponding binary random variable M_i) , then $\mathbf{m} = \{m_i\}_{i=1:n}$ (and $\mathbf{M} = \{M_i\}_{i=1:n}$) , where n is the number of grid cells used to represent the world.

Unfortunately, this choice of representation results in a space of maps that grows exponentially with the number of cells. However, most mapping algorithms make a further assumption of statistical independence between each binary random variable. Due to this assumption, one may compute the posterior belief over the space of maps $\mathcal{M}$ as:

$$p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}) = \prod_{i=1}^{n} p(M_i = m_i \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}), \tag{2.1}$$

leaving us to focus on the simpler and more tractable task of estimating $p(M_i = m_i \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t})$. Further, we make the simplifying assumption that the state of the world at

any time t only depends on data obtained at time t. This is a reasonable assumption, given rich enough sensor and mobile agent information at time t. As such, we may write that:

$$p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}) = \prod_{i=1}^{n} p(M_i = m_i \mid X_t = x_t, Z_t = z_t). \tag{2.2}$$

To compute $p(M_i = m_i \mid X_t = x_t, Z_t = z_t)$, we make use of the mapping algorithm presented by Thrun et al., [106]. Occupancy grids are typically used for mapping in static environments. The application of our work focuses on non-static environments, including moving vehicles and pedestrians. Consequently, we modify the traditional mapping algorithm by removing the time dependence across maps, thus taking a one shot approach with no prior knowledge of the environment. However, if some domain specific knowledge about the dynamics of the environment is known, it can be incorporated into $p(M_i = m_i \mid X_t = x_t, Z_t = z_t)$ through a transition function that links the previous state to the current state.

Landmark Representation

When the environment is represented as a collection of sparse landmarks, the world can be viewed as a collection of salient points in the environment (e.g., people, vehicles, key buildings and natural objects). These salient points are termed landmarks, and the mapping task is to estimate the state of these landmarks given data obtained from sensors. Typically, the state of most interest is the pose of these landmarks. This approach represents the map $\mathbf{m}$ as a collection of k landmarks $\{\xi_i\}_{i=1:k}$ (with corresponding random variables $\{\Xi_i\}_{i=1:k}$), so that now, $\mathbf{m} = \{\xi_i\}_{i=1:k}$ (and $\mathbf{M} = \{\Xi_i\}_{i=1:k}$).

In this work, we have chosen to use pedestrians as landmarks. The state of interest is their position on the 2D floor plane. Concretely, $\xi_i \in \mathbb{R}^2$. One could make use of a Kalman filter to estimate $p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t})$, but the publicly available dataset this model was tested on did not contain enough information to do this. Alternatively, since we are interested in modeling the position of the pedestrian in cases where they are occluded, we have assumed a uniform distribution for $p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t})$. This simple approach represents the fact that when the pedestrian is occluded, we may have no information about its possible location due to the unpredictability of human behavior.

Integrating Humans in Mapping

One of the main contributions of this chapter is the use of human models as a source of sensor information. We argue that the actions of intelligent agents, specifically other drivers in this scenario, are a ubiquitous source of rich information that should not be ignored. However, it is also important that this information be incorporated appropriately with other sources of information, since human agents are highly uncertain and are difficult to model.

Given data on human driver behaviors a, from a set $\mathcal{A}$, and corresponding random variable A, we may reformulate the mapping problem as estimating $p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}, A_{1:t} = a_{1:t})$, where $a_{1:t}$ is a sequence of observed data on human drivers up until time t,

and $A_{1:t}$ the set of corresponding random variables. We will subsequently refer to this human behavior data as an action.

While this idea is indeed general, we restrict ourselves to the case where we only observe actions from the closest driver in front of our ego vehicle. In future work, we will extend formulation to the scenario with multiple driving agents and lanes.

Building on the formulation discussed in the previous section, we estimate $p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}, A_{1:t} = a_{1:t})$ by using Bayes' rule to fuse information obtained from driver data with our map estimate obtained using (2.1). Thus, we have:

$$p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}, A_{1:t} = a_{1:t}) = \frac{p(A_{1:t} = a_{1:t} \mid \mathbf{M} = \mathbf{m}, X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}) p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t})}{p(A_{1:t} = a_{1:t} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t})} . \quad (2.3)$$

As before, we assume that the state of the world at any time t only depends on data obtained at time t. Though this assumption may seem restrictive, in practice, we use actions obtained at time t that contain a history of observed behavior.

We also assume that given a representation of the world, the behavior of the human driver does not depend on the pose of our mobile agent or our sensor information. While this generally might not be true, in the specific context of our application, this assumption is valid due to the relative positioning of the agents. Taking into account the recursive influences is left as future work. Given these assumptions, what we seek to estimate is:

$$p(\mathbf{M} = \mathbf{m} \mid X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}, A_{1:t} = a_{1:t}) = \frac{p(A_t = a_t \mid \mathbf{M} = \mathbf{m}) p(\mathbf{M} = \mathbf{m} \mid X_t = x_t, Z_t = z_t)}{p(A_t = a_t)} \quad (2.4)$$

In the case where the world is represented as an occupancy grid, we have used a driver model $p(a_t \mid m_i)$ that depends on each grid cell and fused the information from the driver to estimate:

$$p(M_i = m_i \mid X_t = x_t, Z_t = z_t, A_t = a_t) = \frac{p(A_t = a_t \mid M_i = m_i) p(M_i = m_i \mid X_t = x_t, Z_t = z_t)}{p(A_t = a_t)} . \quad (2.5)$$

We then make use of (2.2) to obtain $p(\mathbf{M} = \mathbf{m} \mid X_t = x_t, Z_t = z_t, A_t = a_t)$. The next section explains in detail how we obtain the driver model $p(A_t = a_t \mid \mathbf{M} = \mathbf{m})$.

Sensor Models for Drivers

To model the driver as a sensor, we must learn a function that takes in map data and outputs a probability distribution over actions that the driver may take. Our proposed framework is general enough to handle both the occupancy grid and the landmark formulation, depending on the type of driving data used to learn the sensor model.

We evoke concepts from discrete choice theory, a tool from economics that aims to describe, explain, and predict choices between discrete alternatives [109]. We assume we have a discrete set of actions that is both *exclusive* and *exhaustive*, meaning that the driver must pick one and only one of the defined actions.

Likelihood of Actions from Occupancy Grids

Supposing we have a finite collection of driver actions $\mathcal{A}$ and the assumption that each cell in the grid is an independent Bernoulli random variable, we can approximate the probability of an action given the state of cell m_i empirically. For each action, we denote this empirical distribution as $\hat{p}_{N_s}(A = a \mid M_i = m_i)$, where N_s is the total number of trials and a is the action in set $\mathcal{A}$.

We employ this method on simulation dataset from which we are able to extract the relative position of the human driven vehicle to the crosswalk along with its velocity and acceleration information. We seek to learn the distribution over this data given the current map.

Given that learning the high-dimensional distribution is computationally difficult and data intensive, we make a few simplifications to the problem. Rather than learn over the space of all positions, velocities and accelerations, we instead define a finite collection of representative samples of this space observed during experiments.

To determine what these samples should be, we cluster over the data containing the current distance between the human driven vehicle and the crosswalk, and ten evenly spaced samples of both velocity and acceleration over the last half second. We use k-means clustering algorithm to identify k natural groupings in the data. These can be thought of as "actionlets" that correspond to typical sequences of driver behaviors. We then define these clusters as the set of possible actions we may observe. In this work, we make set $k = 10$ as prescribed by grid search.

In doing this, we can easily learn the probability distribution over actions given map data, giving us an approximation of $\hat{p}_{N_s}(A_t = a_t \mid M_i = m_i)$.

Likelihood of Action from Landmarks

Using the landmark interpretation of the mapping problem, the sensor model of the driver must be approximated as the probability of an action given the position of the landmark obstacle. To do this, we apply the logit model from discrete choice theory to find this mapping [109].

Previous work has demonstrated that this method can determine driver actions and intent with high accuracy [31]. This approach employs the EM algorithm to iteratively find the optimal linear combination of features in the dataset to estimate the probability of an action given some map configuration: $p(A_t = a_t \mid \mathbf{M} = \mathbf{m})$.

We employ this method on a real-world dataset. While this dataset does not provide access to the vehicle state information, it alternatively provides descriptions of the ego vehicle's velocity profile that are consistent with actions used in the literature [62]:

1. *Moving Fast:* The vehicle is moving at a speed above predetermined threshold.
2. *Moving Slow:* The vehicle is moving at a speed below predetermined threshold.
3. *Accelerating:* The vehicle increasing its speed.
4. *Decelerating:* The vehicle is decreasing its speed.
5. *Stopped:* The vehicle is stopped.

We make use of these descriptions as actions. Given that these actions inherently take into account time (e.g., moving fast indicates a high constant velocity for a period of time) we do not consider the the sequences of actions over time and only estimate the map given the last observed action.

2.3 Case 1: Occupancy Grid Formulation

We first evaluate our conceptual framework on the map representation of occupancy grids. In this test case, we carry out a user study to collect ground truth information about the state of the world, which is easily translated into the discretized space of occupancy grids.

Experimental Setup

In order to build the driver model for mapping purposes, training, testing, and validation driving data is required. For the scenario considered, there are few publicly available datasets that provide the quality of data required for mapping and driver modeling purposes. Section 2.4 presents the formulation and results on one of these real-world datasets.

Due to lack of available data with full information about the vehicle and environment states, a new dataset was collected to study driver pedestrian interaction. Driver data was collected using PreScan, an industry standard simulation tool that provides vehicle dynamics and customizable driving environments [30]. Using a force feedback steering wheel and pedals for the subject to control the human-driven vehicle, we created various intersection scenarios in which a pedestrian might appear, as shown in Fig. 2.2.

In each trial, the human-driven vehicle began approaching an intersection at an initial distance d_0 and speed v_0. The pedestrian motion was designed to recreate typical pedestrian behaviors. After appearing from behind an occluding obstacle at randomized velocity, the prescribed behaviors included boldly crossing the road, waiting to cross until the approaching vehicle slowed down, and just standing at the side of the road. To discourage anticipating the pedestrian motion, the pedestrian did not appear in half of the instances.

Figure 2.2: Visualization of the driver view from the experiment. As the driver approaches the crosswalk, there is a chance a pedestrian obstacle will appear from behind the bus stop.

Five subjects each completed approximately one hour of experiments. In each trial, the subject was asked to maintain a constant velocity between 10 and 15 mph and stay in their lane, if possible. This resulted in 1,440 example interactions each lasting approximately 5 to 10 seconds, recorded at 30Hz. From this, we generated a total of 281,506 maps to build our sensor models and test our mapping. Twenty percent of this data is used to generate the learned distribution over actions.

For each trial, we collected: the human driven vehicle states and inputs, and the ground truth position of the pedestrian. Using this data, we created a ground truth occupancy for the region in front of the human driver that would be occluded for our ego vehicle. The occlusion is determined using a simple lidar model to determine what the closest obstacles are in the 360° view. We assume only some of the vehicle states are observable from the ego vehicle (i.e., relative position and velocity, distance to crosswalk).

Using the actions defined in Section 2.2, we train a sensor model that maps the ground truth occupancy grid and sensor measurements to a distribution over actions; this human driver model can then be used to impute the occupancy map from observed actions. An example occupancy grid input and the associated action distribution is shown in Fig. 2.3.

To reiterate, we consider a scenario with three agents: the ego vehicle, the human driven vehicle, and the pedestrian, as visualized in Fig. 2.1. The ego vehicle observes the human driven vehicle that is occluding the pedestrian. Based on the observed actions, we construct a posterior belief across possible maps.

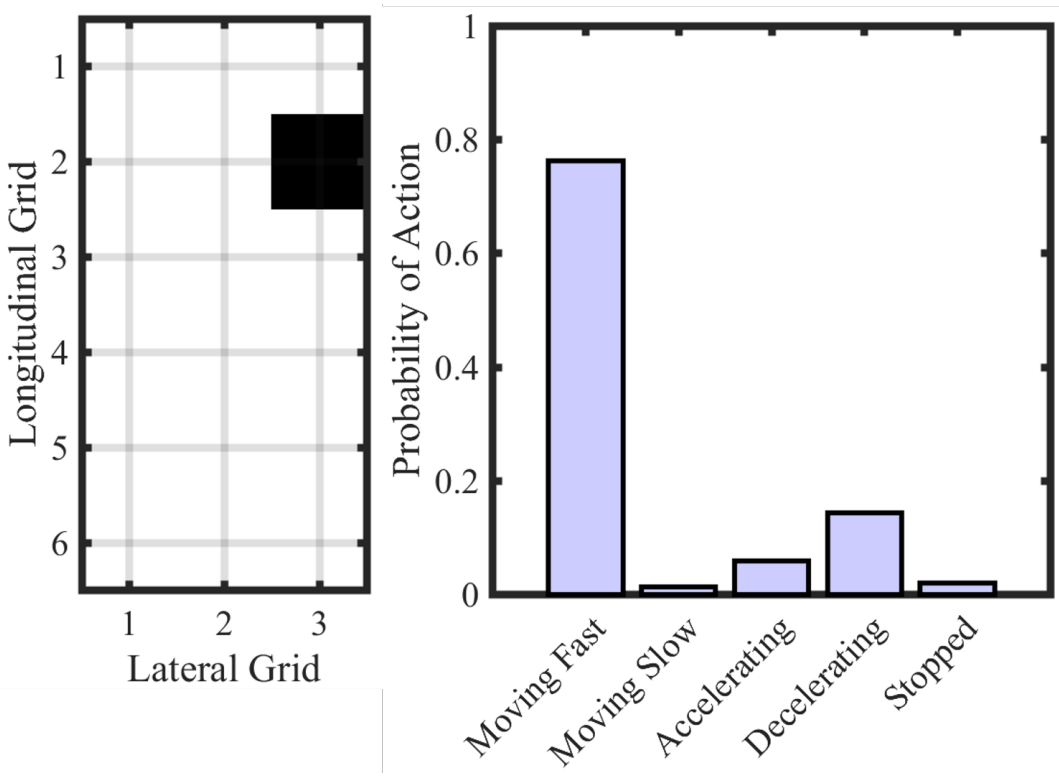

Figure 2.3: Illustrative example of input and output of the driver model. *(Left)* Sample occupancy grid for the region in front of the vehicle (i.e., grid location (2,6) is directly in front of the vehicle), where an occupied space is far ahead and to the right of the vehicle at (3,2). *(Right)* Probability distribution over possible semantic actions given the occupancy map on the left.

Evaluation Metrics

We tested our solution on multiple scenarios from our experimental dataset. As stated previously, each scenario is composed of three agents, an ego vehicle in one lane, a human driven vehicle in the other lane causing the occlusion, and a pedestrian. The ego vehicle was set to follow a constant velocity trajectory behind the human driven vehicle in the second lane (see Fig. 2.1). To evaluate the occupancy grids generated by our approach, we made use of the Image Similarity metric.

The Image Similarity metric, ψ, is used to evaluate the similarity of an occupancy grid to an ideal or ground truth measurement [15]. This metric is computed as:

$$\psi(U,V) = \sum_{c\in\{0,1\}} d(U,V,c) + d(V,U,c) \tag{2.6}$$

where

$$d(U,V,c) = \frac{1}{\#_c(U)} \sum_{U[i]=c} \min \ \{||g(i) - g(j)||_1 : V[j] = c\} \tag{2.7}$$

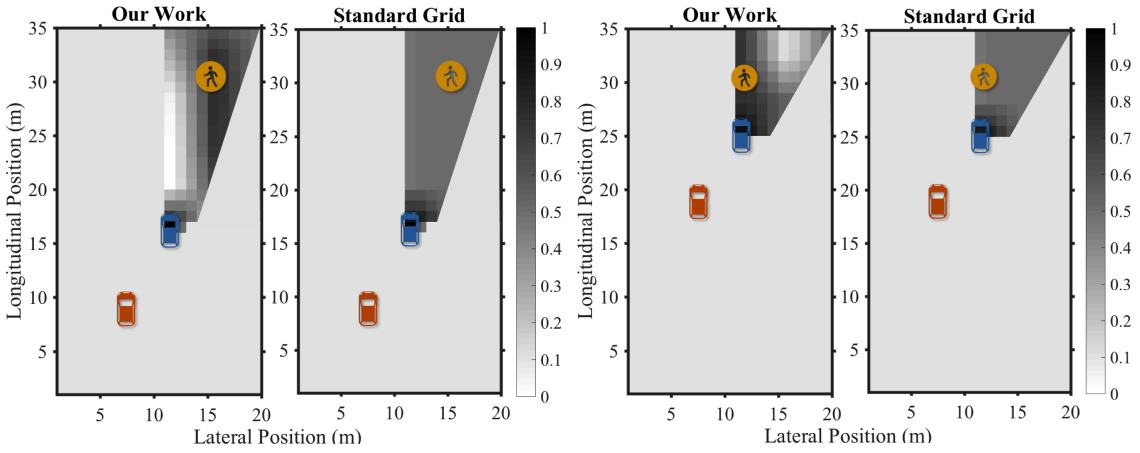

Figure 2.4: Two example comparisons of occupancy grids generated by our method and a standard occupancy grid algorithm. Darker regions indicate greater confidence of occupancy. The orange and blue car icon represent ground truth positions of our ego vehicle and obstructing vehicle, respectively. The pedestrian icon indicates the ground truth position of the pedestrian. In the left image, the obstructing vehicle is observed increasing its velocity while in the right image the obstructing vehicle is observed slowing down.

where $U[i]$ is the occupancy value at grid cell i in map U, $g(\cdot)$ returns the 2D coordinates of grid cell $i, j \in \{1, 2, ..., n\}$, $||\cdot||_1$ gives the Manhattan distance between coordinates, and $\#_c(U)$ is the number of cells in U with occupancy values c.

To make use of ψ, we indicate the occupancy value of each cell by thresholding the probability $p(M_i = 1|X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t})$ as follows:

$$U[i] = \begin{cases} 1 & \text{if } p(M_i = 1|X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}) \geq 0.6 \\ 0 & \text{if } p(M_i = 1|X_{1:t} = x_{1:t}, Z_{1:t} = z_{1:t}) < 0.6 \end{cases}.$$

Results

We compare our results to a standard occupancy grid mapping algorithm that does not incorporate information from the actions of other drivers and to ground truth measurements. We refer to the results from the standard occupancy grid algorithm as "Standard Grid."

Fig. 2.4 shows sample results based on using occupancy grids to represent the environment. The orange and blue vehicle icons indicate the ground truth positions of the ego vehicle and the human-driven vehicle respectively, while the pedestrian icon indicates the ground truth position of the pedestrian. In these scenarios, the pedestrian is occluded from the view of the ego vehicle by the human-driven vehicle in the scene. Consequently, there is large uncertainty concerning the position of the pedestrian using the Standard Grid. Although we do not observe the pedestrian, we observe the behavior of the human-driven vehicle. By

incorporating this information, our driver model is able to help us reason about the likely positions of the pedestrian. Our algorithm can reduce the uncertainty present and provide a more accurate prediction about the position of the pedestrian.

Quantitatively, we applied the metric presented in the previous subsection to compare our solution to the Standard Occupancy Grid. Table 2.1 and Fig. 2.5 show the average scores under the Image Similarity metric, where $t = 0$ indicates the beginning of the trials, $t = T/2$ indicates the middle of each scenario, and and T indicates the end of each scenario. The mean and standard deviation of the results over time are shown in Fig. 2.5. Our solution does significantly better than the standard occupancy grid approach.[1]

Table 2.1: Image Similarity Results for Occupancy Grid Approach.

	Avg.	$t = 0$	$t = T/2$	$t = T$
Standard grid	1.085	1.863	1.071	0.377
Our work	**0.169**	**0.218**	**0.068**	**0.289**

2.4 Case 2: Landmarks in Real-World Dataset

We attempted to assess our framework on a more realistic scenario by testing on a real-world dataset for pedestrian interaction. Ideally, we would use a dataset with a substantial amount of pedestrian interactions, ground truth estimates of vehicle and pedestrian locations over time, and annotations of driver actions. Many of the public datasets do not meet these requirements. The Joint Attention in Autonomous Driving (JAAD) dataset centers on driver-pedestrian interaction, but provides only partial information about the state of the world through semantic action labels and approximate position estimates, making occupancy grids difficult to consider without substantial assumptions [83, 84]. Taking these restrictions into account, we modify our mapping pipeline to one that estimates landmarks that may be occluded, and demonstrate that our framework can be applied to many different settings if context can be taken into account.

JAAD Dataset of Pedestrian Interactions

We use the JAAD dataset, which consists of 346 high-resolution video clips, lasting approximately 5 to 10 seconds each, that are representative of possible crosswalk scenes that often occur in urban driving. These clips are annotated, providing labels associated with the driver and pedestrian actions as well as bounding boxes of detected pedestrians [83, 84]. No vehicle state information (e.g., position, speed, acceleration) is provided with this dataset.

[1]We note that while these results show results overall drivers in the study, we also examined individual driver models. The individual metrics exhibited similar trends to the overall metrics, except for two drivers which had significantly better performance near $t = T$.

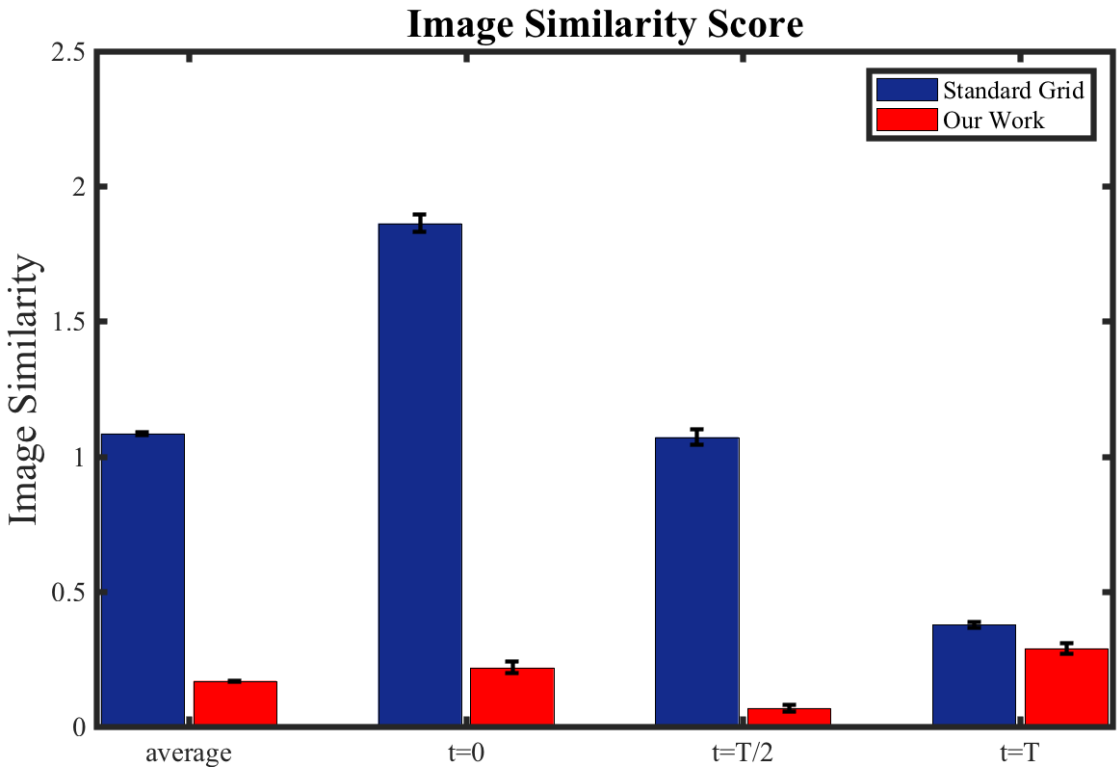

Figure 2.5: Comparison of Image Similarity scores between the occupancy grid generated by our solution and the standard approaches. Lower scores imply better matching. Plots show the mean for the two methods along with the standard error. Our method exhibits significant improvement to the standard occupancy map.

From the pedestrian's bounding box, we estimate the person's position relative to the vehicle camera housing to generate an approximate map for each frame. Given the assumptions required to get this estimate, we assume a Gaussian distribution over our estimates, making this partial, noisy data more inline with the landmark philosophy.

From this dataset, we extract a total of 76,514 samples to train and test our model from. The logistic regression model is trained on 20% of the samples to find the relationship between the action and relative position. An example image and map data are shown in Fig. 2.6.

Results

Using the driver model learned from the JAAD dataset, we predict the location of the pedestrian as a landmark, as described in Section 2.2. We assume an uninformed (i.e., uniform) prior over the occluded space, and show how the output of our algorithm provides

Figure 2.6: Example image from JAAD Dataset with the pedestrian in a labeled blue bounding box [83].

a posterior distribution conditioned on the human driver actions that can improve the estimation of the pedestrian's location. To evaluate the improvement that the map generated by our landmarks model of the environment provides, we compare the likelihood of the pedestrian's true location in the posterior distribution to the prior distribution.

The results of our approach compared with the uniform prior are shown in Table 2.2. Fig. 2.7 presents a sample output of the our solution using the landmark representation and that of the uniform prior. The orange and blue vehicle icons represent the ground truth positions of the ego vehicle and the human-driven vehicle respectively, while the pedestrian icon represents the ground truth position of the pedestrian.

Once again, in this scenario, the pedestrian is occluded from the view of the ego vehicle. The plots in the figure represent the estimated posterior density of the position of the pedestrian, with darker regions indicating higher density values. As shown, by incorporating the driver model learned from data, our solution is able to predict the likely position of the pedestrian during occlusions.

Our method provides useful predictions in a majority of the actions and is most informative in safety critical situations. The scenarios where our methodology is less informative are intuitive if we consider how drivers behave in the real-world. When we drive and observe other drivers maintaining a constant speed, we gain little insight about occluded obstacles. We observe from the driver model derived from the JAAD dataset that the two constant velocity actions (moving fast and slow) are not informative without detailed contextual information.

Further, since we partition the dataset to only consider samples where the pedestrian might be occluded, these two actions are underrepresented relative to the other labels. Because of these points, our approach only exhibits improved performance on a subset of the actions. We are of the opinion that with ground truth position information and more evenly distributed data, significantly better improvements can be obtained as was the case in the simulated environment.

2.5 Interpretation as an Estimation Problem

The presentation in this chapter will have hinted to the fact that at its core, the problem presented is an estimation problem. Explicitly, we note that the parameter(s) to be estimated is the map of the environment $\mathbf{M}$. The likelihood function $p(A_{1:t} \mid \mathbf{M})$ is the data driven driver model and the prior distribution $p(\mathbf{M} \mid X_{1:t}, Z_{1:t}, A_{1:t})$ is obtained using a standard occupancy grid mapping algorithm [106]. A data-driven likelihood function is advantageous because it is otherwise difficult to relate the behavior of human drivers to the state of the environment.

In this chapter, rather than seeking the Maximum A Posteriori (MAP) estimate, we instead compute the posterior distribution over the space of maps. The benefit being that it gives us more information about our parameter of interest than just the MAP estimate.

2.6 Discussion

By exploiting the actions of other intelligent agents, a great deal of information can be inferred about the environment. We have presented a methodology that uses driver models as sensors to impute maps that can be used to improve planning in the face of uncertainty. Thus, regions of the map that would otherwise be occluded can be imputed, providing an estimation of the environment's state. We validate this concept on two different map representations and datasets, demonstrating significantly improved performance over standard mapping techniques.

Table 2.2: Evaluation Metric on JAAD Dataset showing the probability of observing the pedestrian at ground truth location for each action.

Action	Uniform Prior	Our Work	Improvement Ratio
Moving Fast	**0.064**	0.002	–0.963
Moving Slow	**0.064**	0.027	–0.569
Accelerating	0.064	**0.067**	0.049
Decelerating	0.064	**0.080**	0.242
Stopped	0.064	**0.257**	3.040

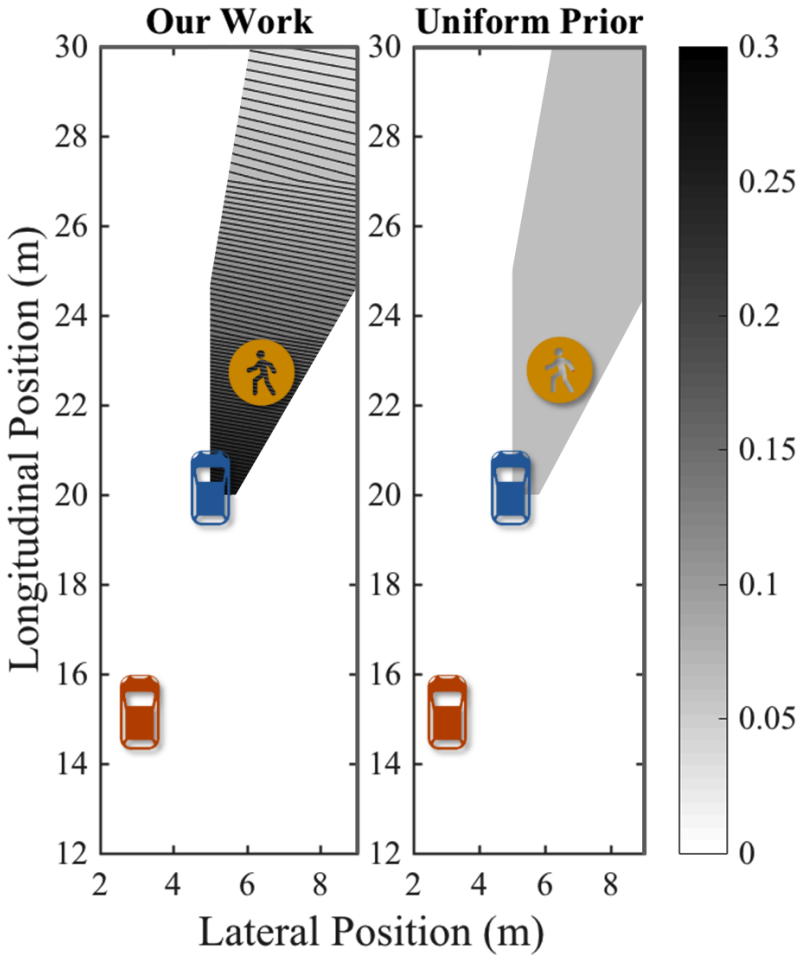

Figure 2.7: Example comparison of landmark map generated by our method versus using a uniform prior over possible locations, where the obstructing vehicle is *stopped*. White space indicates un-occluded regions. Darker regions indicate greater belief in the pedestrian position. The orange car, blue car, and pedestrian icons represent ground truth positions of our ego vehicle, obstructing vehicle, and pedestrian, respectively.

Chapter 3

Energy: Energy Disaggregation

List of Symbols

$[D]$	The set $\{1, 2, \ldots, D\}$.
$\mathcal{U}^{(i)}$	Random vector representing the usage pattern of device i over the time horizon.
$\mathcal{U}$	Random matrix whose i^{th} column is $\mathcal{U}^{(i)}$.
$\mathcal{Y}^{(i)}$	Random vector representing the energy consumption of device i over the time horizon.
$\mathcal{Z}$	Aggregate energy consumption signal, the sum of $\mathcal{Y}^{(i)}$ over all devices.
s_n	The n^{th} time segment, i.e. the interval $\{t_n, t_n + 1..., t_{n+1} - 1\}$.
$a[s_n]$	For some signal a, the value of a over the n^{th} time segment.
$a[t_1 : t_n]$	For some signal a, the value of a from time t_1 to t_n.
$\mathcal{F}$	The set of filters in the filter bank.
$\mathcal{H}^{(i)}$	The device model for device i.
$\mathcal{H}$	The aggregate device model, the sum of $\mathcal{H}^{(i)}$ over all devices.

In the previous chapter we developed an algorithm for estimating the state of the environment in an autonomous driving scenario. We now turn our attention to a different but equally important problem of energy disaggregation. We present a data-driven perception algorithm for energy disaggregation developed in this chapter and the corresponding Appendix. Here, perception is the process of turning raw aggregate energy consumption data into meaningful energy consumption estimates for the individual appliances in a building.

3.1 Introduction

Concerns about the earth's rising temperature and other effects of global warming have prompted a worldwide response to reducing energy consumption and greenhouse gas emissions. It has been suggested that reducing the energy consumption of residential and commercial buildings can play a significant role in curbing energy consumption and greenhouse gas emissions [20, 63]. In fact, in the U.S. alone, residential and commercial buildings currently account for about 40% of total energy consumption and greenhouse gas emissions [4, 112]. This energy consumption is usually reported to consumers in aggregated form through a monthly bill. However, studies [7] have shown that feedback on appliance level consumption patterns should help reduce energy consumption better than feedback through an aggregated energy bill.

Presently, the energy grid is set up to allow the measurement of aggregate consumption data. Installing plug level sensors to measure the energy consumption of each appliance is cumbersome and intrusive, in the sense that it reduces consumers' perceived privacy. In addition, modifying the energy grid with high frequency, high resolution, metering and data transfer technology for multiple devices per household would be cost prohibitive. Energy disaggregation or Non-Intrusive load monitoring (NILM), which refers to the process of estimating appliance level energy consumption patterns given aggregated energy consumption measurements, presents a way to obtain fine-grained device level measurements without incurring huge infrastructural and privacy costs.

Furthermore, disaggregation allows for better control and modeling of energy systems. The increased granularity of information presented through disaggregation can help in creating better models for energy systems which in turn help in detecting anomalous behavior. This fault detection coupled with better modeling presents an opportunity for finer control of critical energy systems such as heating, ventilation, and air conditioning (HVAC) units.

Utility companies may also benefit from energy disaggregation. Disaggregated data may be used to discover user consumption patterns. These user consumption patterns in conjunction with other data may be used for market segmentation. Market segmentation allows utility companies more accurately target consumer groups with products that would be beneficial to them, thereby making more efficient use of product development funds and improving customer retention.

Essentially, to do this well and gain the trust of all beneficiaries across the board, we argue that disaggregation algorithms should produce "realistic" looking and "believable" signals. I.e., the results should not only be on average similar to the ground truth, but should follow a similar form and pattern to the ground truth signal. Good device models, which may sometimes take sophisticated forms with complex temporal dependencies, are crucial to obtaining such realistic estimates. In addition, it may be necessary to incorporate sophisticated priors on the behavior of devices.

With many approaches to disaggregation, it can be difficult to incorporate such sophisticated models and priors and perform inference efficiently. For example, ensuring temporal

similarity for predicted values in Hidden Markov Models and their variants can lead to an exponential increase in the size of the state space.

Unfortunately, many disaggregation metrics do not strongly emphasize the "believability/realism" of the predicted signal and as such, an algorithm may score highly on popular metrics but not look like the signal to be predicted.

What we present in this chapter is a framework that allows the flexibility in device modeling required to produce signals more similar in pattern to the those produced by the actual devices, while intelligently pruning wasteful computation in the optimization process necessary to produce an optimal disaggregation. We test an implementation of our framework on a public dataset and show that we achieve results comparable to baselines, while achieving more realistic disaggregation trajectories even with a very simple but more flexible device model. We present the energy disaggregation problem in an adpative filter banks framework and expand on our previous formulation that combines the filter bank idea with linear dynamical models of devices. In particular, we provide a broader formulation of the framework and revisit assumptions necessary for optimality.

This chapter is organized as follows. In Section 3.2, we give an overview of previous work on energy disaggregation. In Section 3.3, we formally define the problem of energy disaggregation. Section 3.4 introduces our framework and proposed algorithm for disaggregation. We prove properties of this algorithm in Section 3.5, and give implementation details and experimental results in Sections 3.6, B.1 and 3.8.

3.2 Background

Energy disaggregation is a well studied problem. Solutions to the problem of energy disaggregation include the use of hardware devices such as smart plugs to measure device-level consumption. As stated earlier, these solutions can be somewhat intrusive. Rather, the set of solutions we consider here fall under the category of non-intrusive algorithmic solutions that make use of aggregate whole building energy measurements. The works in [13, 12, 121] provide in depth surveys on the breadth of solutions offered in the literature. Yet, in an effort to be as accessible and self-contained as possible, we give an overview of some of these solutions and highlight the similarities and differences between our solution and the existing work.

Broadly, energy disaggregation algorithms can be split into supervised and unsupervised methods. Supervised disaggregation algorithms are typically discriminative algorithms that require the use of a labelled training dataset containing signatures or features associated to an appliance type. The data for each appliance may be collected and labelled using smart plugs or may be annotated through the use of smartphone technologies [115]. This tends to be a cumbersome process. Fortunately, there exist datasets such as [70, 60, 6, 55] that allow the researcher get access to good data without having to set up all the processes necessary to collect it.

Dissimilarly, unsupervised methods do not require labelled training data for the constituent devices in the aggregate signal. Nonetheless, they may require parameter tuning to ensure good performance and generalization.

Examples of current supervised approaches include optimization-based methods [61, 35, 100], change detection and clustering-based approaches [29, 82] and pattern recognition methods [36, 54]. In [61] the authors make use of sparse coding to reconstruct the aggregate signal from a library of signatures. In [35] the authors also make use of sparse coding but incorporate priors for temporal smoothness, device sparsity and co-occurence. They also represent each device using a mixture of dynamical systems. Our work is similar to these methods in the sense that we also seek to find the most likely combination of signatures from a library of device signatures. Unlike these works, the optimization procedure in our work makes use of a technique similar to Branch and Bound. We also make no explicit sparsity requirement. More recently, there has been a body of work such as [54, 118, 72, 64], making use of artificial neural networks (ANNs) for the task of energy disaggregation. Our work differs from these in that we do not learn an optimal set of weights to make good predictions, but rather reason over how, given a model of devices, the states of each device contribute to the optimal disaggregation. However, our framework may also benefit from using ANNs for device modeling.

Examples of unsupervised methods include methods based on factorial Hidden Markov Models (HMMs) [59, 57, 79], difference Hidden Markov Models [77], Hierarchical Dirichlet Process Hidden semi-Markov Models [50], and the use of temporal ordering to uncover motifs [91]. Most unsupervised methods do not require the use of device signatures but make the assumption of piecewise constant power consumption. However, the authors in [71] introduce an HMM with superstates and a sparse Viterbi algorithm that learns device model parameters from training data. Examples of unsupervised methods not making use of HMMs include [65]. Here, users' usage behavior is modeled using a Marked Hawkes Process and incorporated into the disaggregation process.

In our previous work [26], we introduced a filter bank approach to energy disaggregation and combine this with a dynamical system model of devices. The filter bank framework is similar to HMM frameworks in the sense that both methods essentially formulate hypotheses regarding which devices are on at each time instant. However, unlike HMMs, we made use of dynamical models to help identify devices. In this chapter, we expand on our previous effort by providing a broader framework to reason about the algorithm, strengthening proofs and providing more precise assumptions under which optimality is guaranteed. In addition, this framework removes the restriction of a finite impulse response (FIR) model of devices. We also test our work on a public dataset and compare our performance to two other publicly available implementations of energy disaggregation algorithms. We show that we achieve more robust results than the baselines and more realistic disaggregation trajectories.

This chapter is part of work done with the co-author in [25, 27]. Some of the theoretical formulation presented here are also explained in [25], but we present it again here so that the chapter is self contained. One of the main additional contributions of this chapter is to extend the implementation to large public data as well as show improved methods for device

modeling as contained in the Appendix to this chapter.

3.3 Problem Formulation

Notation and concepts

In this section, we outline a formalization of the basic components that are present in our energy disaggregation problem. There are $D \in \mathbb{N}$ devices, and we work with a time horizon of $T \in \mathbb{N}$ discrete time steps. For $i \in [D]$, $\mathcal{U}^{(i)}$ is a random vector taking on values $u^{(i)} \in \mathbb{R}^{T+1}$, where $u^{(i)}$ is a realization of the usage/ input pattern of device i over the time horizon. Similarly, $\mathcal{Y}^{(i)}$ is a random vector taking on values $y^{(i)} \in \mathbb{R}^{T+1}$, where $y^{(i)}$ is a realization of the energy consumption of device i. Finally, let $\mathcal{U}$ be a random matrix whose i^{th} column is $\mathcal{U}^{(i)}$, so that $\mathcal{U}$ takes on values $u \in \mathbb{R}^{(T+1)\times D}$.

Assumption 3.3.1. (Conditional independence of usage patterns) *We assume that the energy consumption signals follow a known distribution*

$$\mathcal{Y}^{(i)}|\mathcal{U}^{(i)} \sim P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}(y^{(i)}|u^{(i)}) \ .$$

We also assume that:

$$\mathcal{U} \sim P_{\mathcal{U}}(u) \ .$$

Additionally, the energy consumption of device i, $\mathcal{Y}^{(i)}$, is assumed to be conditionally independent of the usage patterns of other devices $\mathcal{U}^{(j)}$ for $j \neq i$, given $\mathcal{U}^{(i)}$. The aggregate energy consumption, which is known, is given by $\mathcal{Z} = \sum_{i\in[D]} \mathcal{Y}^{(i)}$. So that,

$$P_{\mathcal{Z}|\mathcal{U}}(z|u) =$$

$$\int \cdots \int \left[\left(\prod_{i=1}^{D-1} P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}(y^{(i)}|u^{(i)}) \right) P_{\mathcal{Y}^{(D)}|\mathcal{U}^{(D)}} \left(z - \sum_{i=1}^{D-1} y^{(i)} \middle| u^{(D)} \right) \right] dy^{(1)} \ldots dy^{(D-1)} \tag{3.1}$$

$$= (P_{\mathcal{Y}^{(1)}|\mathcal{U}^{(1)}} * \ldots * P_{\mathcal{Y}^{(D)}|\mathcal{U}^{(D)}})(z|u) \ ,$$

where $*$ is the convolution operator.

Then, formally, we can define the Bayesian energy disaggregation problem as the tuple[1]:

$$(P_{\mathcal{U}}, \{P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}\}_{i\in[D]}, z) \ , \tag{3.2}$$

[1]Note that D is implicitly contained in the size of the set in the first element of the tuple, and T is implicitly contained in the domain of the probability measures.

where the goal is to obtain an estimate for the usage pattern u, given z , $P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}$, and $P_{\mathcal{U}}$.

The *maximum a posteriori (MAP)* estimate of u is given by:

$$\widehat{u_{\text{MAP}}} = \arg\max_{u} \left(P_{\mathcal{U}}(u) P_{\mathcal{Z}|\mathcal{U}}(z|u) \right) \quad . \tag{3.3}$$

Additionally, we will suppose there is a unique element in the arg max, although this assumption is for simplicity of notation and not necessary for the development of the subsequent text[2]. Finally, in this problem formulation, we are given $P_{\mathcal{U}}$ and $P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}$. In practice, these will have to be learned from a training set of individual devices' energy consumption data.

Switching times

In general, finding $\widehat{u_{\text{MAP}}}$ could be very difficult. However, an observation that helps us solve the problem of energy disaggregation is the following: when data is collected at high frequencies, the inputs to devices are often piecewise constant. For example, the Reference Energy Disaggregation Dataset (REDD) [60] contains data sampled at rates faster or equal to 1/3 Hz. In contrast, heating, ventilation, and cooling (HVAC) systems switch states at much slower rates, and energy consumers change lighting settings at slower rates as well. With this observation in mind, we can define the switching times of a given input.

Definition 3.3.1. (Switching times). *For some fixed $v : \{0, 1, \ldots, T\} \to \mathbb{R}^D$, we can define the* switching times *of v, denoted $T_{switch}(v) \subset \{0, 1, \ldots, T\}$, as the unique set such that:*

- $v[t-1] \neq v[t]$ *for all* $t \in T_{switch}(v)$.
- $v[t-1] = v[t]$ *for all* $t \notin T_{switch}(v)$.

We adopt the convention where $v[-1] = 0$.

We will often use the notation $T_{switch}(v) = \{t_1, t_2, \ldots, t_N\}$ with the understanding that N depends on v and $t_1 < t_2 < \cdots < t_N$. Switching times will also be referred to as a segmentation. Each interval $\{t_n, t_n + 1, \ldots, t_{n+1} - 1\}$ for $n \in \{0, 1, \ldots, N\}$ is defined as a segment *s_n, with the convention $t_0 = 0$ and $t_{N+1} = T + 1$. In addition, we adopt the convention that $N = 0$ when $T_{switch} = \emptyset$.*

If we think of the rows of $\mathcal{U}$ as indexed by time, then we can similarly define the switching times of the random variable $\mathcal{U} \sim P_{\mathcal{U}}$, where $T_{switch}(\mathcal{U})$ is now a random set-valued element.

Our observation allows us to think about the energy disaggregation problem as a two step process. First, we find the switching times of $\widehat{u_{\text{MAP}}}$, and subsequently estimate the actual value of $\widehat{u_{\text{MAP}}}$. If we consider every possible switching time, we would solve the energy disaggregation problem exactly. This is formally stated in Proposition 3.3.1.

[2]For full rigor, we can simply replace any statement of the form '$x = \arg\max_{x \in C} f(x)$' with 'pick any $x \in \arg\max_{x \in C} f(x)$'.

Definition 3.3.2. *For any set $T_{switch} \subset \{0, 1, \ldots, T\}$, we define $s(T_{switch}|z)$ as follows:*

$$s(T_{switch}|z) = \max_{\substack{u: \\ T_{switch}(u)=T_{switch}}} \left(P_{\mathcal{U}}(u)P_{\mathcal{Z}|\mathcal{U}}(z|u)\right) \ .^3 \tag{3.4}$$

Proposition 3.3.1. *$T_{switch}(\widehat{u_{\text{MAP}}})$ satisfies the following condition.*

$$T_{switch}(\widehat{u_{\text{MAP}}}) = \arg\max_{T_{switch}} s(T_{switch}|z) \ . \tag{3.5}$$

Additionally, if $T_{switch}(\widehat{u_{\text{MAP}}})$ is known, then:

$$\widehat{u_{\text{MAP}}} = \arg\max_{\substack{u: \\ T_{switch}(u)=T_{switch}(\widehat{u_{\text{MAP}}})}} \left(P_{\mathcal{U}}(u)P_{\mathcal{Z}|\mathcal{U}}(z|u)\right) \ . \tag{3.6}$$

Similarly:

$$P_{\mathcal{U}}(\widehat{u_{\text{MAP}}})P_{\mathcal{Z}|\mathcal{U}}(z|\widehat{u_{\text{MAP}}}) = \max_{T_{switch}} s(T_{switch}|z) \ . \tag{3.7}$$

Proposition 3.3.1 implies that the problem of energy disaggregation can be broken up into two parts. First, we identify the set of switching times $T^*_{switch} = \arg\max_{T_{switch}} s(T_{switch}|z)$. Then, we calculate the MAP estimate $\widehat{u_{\text{MAP}}}$ by maximizing the posterior probability $P_{\mathcal{U}}(u)P_{\mathcal{Z}|\mathcal{U}}(z|u)$ across the set of u such that $T_{switch}(u) = T^*_{switch}$.

Next, we will argue that the optimal $\hat{u}$ for a fixed segmentation is a light calculation, given some assumptions motivated by the energy disaggregation application. Then, we will present an algorithm which allows us to selectively consider possible switching times, while still ensuring the recovery of $\widehat{u_{\text{MAP}}}$.

3.4 Proposed Framework

In this section, we will first introduce conditions under which the estimation of u is computationally inexpensive for a fixed segmentation. Essentially, the next assumption will ensure that the estimation problem in each segment is decoupled. First, we introduce the notation

$$y[s_n] = (y[t_n], y[t_n + 1], \ldots, y[t_{n+1} - 1]) \ . \tag{3.8}$$

Assumption 3.4.1. (Conditional independence of segments) *Given $T_{switch}(u) = \{t_1, t_2, \ldots, t_N\}$, we can express $P_{\mathcal{U}}$ in the following way:*

$$P_{\mathcal{U}}(u) = \prod_{n=0}^{N} P_{\mathcal{U}_{s_n}}(u[t_n]) \ . \tag{3.9}$$

[3] In the most general case, it is possible for the supremum to exist outside the feasible set. However, we will assume this is not the case, since in most implementations the space of inputs is finite and u is quantized.

Note: This product depends on $T_{switch}(u)$, $u[s_n] = \text{constant} = u[t_n]$. $P_{\mathcal{U}_{s_n}}$ are in general non-negative functions indexed by s_n and not necessarily densities; this condition is different from statistical independence.

Additionally, conditioned on $T_{switch}(u) = \{t_1, t_2, \ldots, t_N\}$, the following are independent:

$$\{y^{(i)}[s_0], y^{(i)}[s_1], \ldots, y^{(i)}[s_n]\} \ . \tag{3.10}$$

Therefore, we can write that

$$P_{\mathcal{Z}|\mathcal{U}}(z|u) = \prod_{n=0}^{N} P_{\mathcal{Z}|\mathcal{U}_{s_n}}(z[s_n]|u[t_n]) \ . \tag{3.11}$$

In the context of energy disaggregation, these assumptions have a natural interpretation. The assumption (3.10), imposes a condition on $P_{\mathcal{Z}|\mathcal{U}}$ that the observation at time t only depends on how devices are being used in that segment. The energy consumption of devices generally undergo a transient as the device switches state, and then approach a steady-state energy consumption. Intuitively, what this assumption states is that time between device switches is long enough that devices reach steady-state in each segment. As mentioned before, this assumption is motivated by the fact that the duration of these transients is much shorter than the duration of our usage patterns, e.g. lighting reaches steady-state far more quickly than the time period when light switches stay on or off during normal use. We note that this assumption is likely necessary for good performance: if this assumption were violated, then the transient signatures of devices would collide and interfere with each other.

We note a consequence of Assumption 3.4.1 and Proposition 3.3.1.

Proposition 3.4.1. *Let $T_{switch}(\widehat{u_{\text{MAP}}}) = \{t_1, t_2, \ldots, t_N\}$. Under Assumption 3.4.1:*

$$\widehat{u_{\text{MAP}}} = \arg\max_{u} \sum_{n=0}^{N} \left[\ln\left(P_{\mathcal{U}_{s_n}}(u[t_n])\right) + \ln\left(P_{\mathcal{Z}|\mathcal{U}_{s_n}}(z[s_n]|u[t_n])\right) \right] \ . \tag{3.12}$$

If $T_{switch}(\widehat{u_{\text{MAP}}})$ is given, then these optimizations are decoupled, so:

$$\widehat{u_{\text{MAP}}}[t_n] = \arg\max_{u} \left[\ln\left(P_{\mathcal{U}_{s_n}}(u[t_n])\right) + \ln\left(P_{\mathcal{Z}|\mathcal{U}_{s_n}}(z[s_n]|u[t_n])\right) \right] \ . \tag{3.13}$$

Proposition 3.4.1 states that, if we are given the true switching time $T_{switch}(\widehat{u_{\text{MAP}}})$, then calculating $\widehat{u_{\text{MAP}}}$ is very tractable. In fact, if we are given a collection of potential switching times, calculating $\widehat{u_{\text{MAP}}}$ is still tractable, provided $T_{switch}(\widehat{u_{\text{MAP}}})$ is in the collection and the collection is not too large.

Next, we will develop an algorithm to achieve this: it will selectively consider candidate subsets of $\{0, 1, \ldots, T\}$ such that it considers significantly less than 2^{T+1} candidates, but will still consider $T_{switch}(\widehat{u_{\text{MAP}}})$. We draw on results in the adaptive filtering literature. In our

particular case, we use a filter bank approach to handle the energy disaggregation problem. A filter bank is a collection of filters, and the adaptive element of a filter bank is in the insertion and deletion of filters, as well as the selection of the optimal filter.

As previously mentioned, there are 2^{T+1} different possible segmentations of $\{0, 1, \ldots, T\}$ so iteratively considering each one is intractable. The process of finding the best segmentation can be seen as exploring a binary tree. This is visualized in Fig. 3.1.

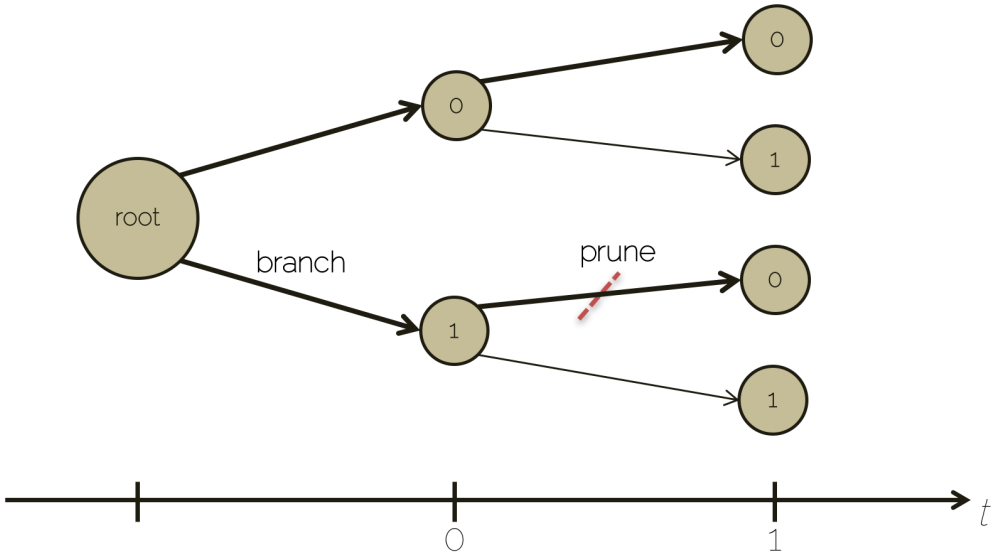

Figure 3.1: A segmentation T_{switch} can be thought of as a leaf node on a binary tree of depth T. That is, T_{switch} corresponds exactly to one leaf node of this binary tree. Additionally, we can associate a node at depth t with a switching time by assuming that no switches happen after t.

Consider how nodes in this tree correspond to segmentations. At depth t, if $t \in T_{switch}$, then we take the 1 branch. If $t \notin T_{switch}$, then we take the 0 branch. In this fashion, each of the 2^{T+1} leaf nodes can be related to a unique segmentation. Additionally, if we pick a node at depth t, we can associate it with a switching time by just taking the 0 branch repeatedly to the leaf, i.e. assuming that no switches occur after time t. Thus, *every* node on this tree can be associated with a switching time.

Before we can introduce our algorithm, we will define one intermediary function which calculates the maximum posterior probability when only given the measurements up until time t, $z[0:t]$.

Definition 3.4.1. *For any* $t \in \{0, 1, \ldots, T\}$, $T_{switch} = \{t_1, t_2, \ldots, t_N\} \subset \{0, 1, \ldots, t-1\}$,

and $z \in \mathbb{R}^{T+1}$*, let* $t_{N+1} = t$ *and let* $T'_{switch} = T_{switch} \cup t_{N+1}$*. Then, we can define:*

$$s_t(T_{switch}|z) = \max_{\widehat{u}:T_{switch}(\widehat{u})=T'_{switch}} \prod_{n=0}^{N} P_{\mathcal{U}_{s_n}}(u[t_n]) P_{\mathcal{Z}|\mathcal{U}_{s_n}}(z[s_n]|u[t_n]) \ . \tag{3.14}$$

Finally, we introduce our algorithm in Algorithm 1. Note that, given z, we can calculate T since $z \in \mathbb{R}^{T+1}$.

Algorithm 1 Energy disaggregation via filter banks

procedure EDFB($z, P_{\mathcal{U}}, P_{\mathcal{Z}|\mathcal{U}}, \mathcal{U}, p_{thres}$)
  $\mathcal{F} \leftarrow \{\emptyset\}$ ▷ Initialize the filter bank with ▷ one filter corresponding to $T_{switch} = \emptyset$.
  for $t \in \{0, 1, \ldots, T\}$ **do**
    for $T_{switch} \in \mathcal{F}$ **do** ▷ Prune the segmentations T_{switch} whose
      if $s_t(T_{switch}|z) < p_{thres}$ **then** ▷ likelihood falls below a certain threshold
        remove T_{switch} **from** $\mathcal{F}$ ▷ when considering data up until time t.
    $T_{list} \leftarrow \arg\max_{T_{switch} \in \mathcal{F}} s_t(T_{switch}|z)$ ▷ Branch the segmentations T_{switch}
    for $T_{switch} = \{t_1, t_2, \ldots, t_N\} \in T_{list}$ **do** ▷ with the highest ▷ likelihood up to time t.
      $t_{N+1} \leftarrow t$
      append $T_{switch} \cup t_{N+1}$ **to** $\mathcal{F}$
  pick any $T_{switch} \in \arg\max_{T_{switch} \in \mathcal{F}} s(T_{switch}|z)$ ▷ Find a set of
  pick any $u^* \in \arg\max_{\widehat{u}:T_{switch}(\widehat{u})=T_{switch}} P_{\mathcal{U}}(\widehat{u}) P_{\mathcal{Z}|\mathcal{U}}(z|\widehat{u})$ ▷ optimal switching times,
  return u^* ▷ and calculate the u^* value ▷ conditioned on any one of the optimal switching times.

As previously hinted, our algorithm selectively explores branches of a binary tree. Limiting the growth of the filter bank $\mathcal{F}$ can be done by deciding which branches to expand and which branches to prune. This sort of formulation lends itself very easily to an online formulation of the filter banks algorithm. In fact, it is more intuitive to think of the algorithm in an online fashion.

At time t, by default, we only follow the 0 branch. For example, in Fig. 3.1, this corresponds to following only the top path, as done at the [root→0] node. We choose to branch a filter, i.e. explore both the 0 and 1 branches on the binary tree, only if it corresponds to one of the most likely segmentations. As an example, this is done at the [root] node in Fig. 3.1. The branching corresponds to adding the bottom path to the [root] node (creating the [root→1] node) as well as the top path (creating the [root→0] node). At the beginning of $t = 1$, the filter bank will contain $T_{switch} = \{0\}$ and $T_{switch} = \{\}$.

Additionally, at time t, we prune any paths that have sufficiently low likelihood. That is, we remove the segmentation T_{switch} from $\mathcal{F}$ if $s_t(T_{switch}|z) < p_{thres}$, where p_{thres} is an algorithm parameter. This is depicted by the dotted line in Fig. 3.1; this involves removing $T_{switch} = \{0\}$ from our filter bank when $t = 1$. After pruning, none of this node's children will be explored, so no node beginning with [root→1] will be present in the filter bank after time $t = 2$.

Thus, Algorithm 1 will only consider a subset of the leaf nodes of the binary tree. However, by intelligently deciding which subset to consider, we can maintain good performance of energy disaggregation algorithm. In fact, Algorithm 1 will recover an MAP estimate of the input under the conditions presented in the previous section. We will prove this formally below.

3.5 Theory

Theorem 3.5.1. (Optimality of energy disaggregation via filter banks without pruning) *Under Assumption 3.4.1, for $p_{thres} = 0$, the u^* returned by Algorithm 1 is a MAP estimate for the energy disaggregation problem $(P_{\mathcal{U}}, P_{\mathcal{Z}|\mathcal{U}}, z)$.*

Intuitively, this proof works as follows. Suppose an MAP estimate were removed from the binary tree due to selective branching. This means that, at a particular point in time, there is a switching time t where the algorithm did not branch the correct segmentation. However, at time t, the algorithm did branch some segmentation T_{switch}, which was optimal based on the observations up until time t. By the decoupling in Proposition 3.4.1, we can improve our performance by replacing the MAP estimate's switching times up until time t with T_{switch} instead. Thus, a MAP estimate was not removed. The formal proof follows.

Proof. Suppose not, i.e. there exists an MAP estimate $\tilde{u}$ such that:

$$P_{\mathcal{U}}(\tilde{u})P_{\mathcal{Z}|\mathcal{U}}(z|\tilde{u}) > P_{\mathcal{U}}(u^*)P_{\mathcal{Z}|\mathcal{U}}(z|u^*),$$

and

$$P_{\mathcal{U}}(\tilde{u})P_{\mathcal{Z}|\mathcal{U}}(z|\tilde{u}) \geq P_{\mathcal{U}}(u)P_{\mathcal{Z}|\mathcal{U}}(z|u)$$

for any u.

By the construction of the algorithm, there exists a time $t \in T_{switch}(\tilde{u})$ such that Algorithm 1 did not branch the node corresponding to $T_{switch}(\tilde{u}) \cap \{0, 1, \ldots, t-1\}$.

Now pick a T_{switch} such that $T_{switch} \cap \{0, 1, \ldots, t-1\}$ was branched at time t. It follows that $s_t(T_{switch}|z) > s_t(T_{switch}(\tilde{u})|z)$.

Let

$$T'_{switch} = (T_{switch} \cap \{0, 1, \ldots, t-1\}) \cup (T_{switch}(\tilde{u}) \cap \{t, t+1, \ldots, T\}),$$

i.e. T'_{switch} takes the switching times of T_{switch} prior to t and the switching times of $\tilde{u}$ after t.

By the decoupling noted in Proposition 3.4.1, we have for any $\tau > t$:

$$s_\tau(T'_{switch} \cap \{0, 1, \ldots, \tau-1\}|z) > s_\tau(T_{switch}(\tilde{u}) \cap \{0, 1, \ldots, \tau-1\}|z) \ .$$

This also includes the case where $\tau = T + 1$. That is:

$$s(T'_{switch}|z) > s(T_{switch}(\tilde{u})|z) \ .$$

However, by Proposition 3.3.1, we can conclude that $\tilde{u}$ is in fact not an MAP estimate. This contradiction allows us to happily conclude that u^* is, in fact, an MAP estimate. □

Corollary 3.5.1.1. (Optimality of energy disaggregation via filter banks) *In addition to Assumption 3.4.1, suppose there exists an MAP estimate $\tilde{u}$ such that, for all $t \in \{0, 1, \ldots, T\}$:*

$$s_t(T_{switch}(\tilde{u}) \cap \{0, 1, \ldots, t-1\}|z) \geq p_{thres} \ . \tag{3.15}$$

Then, Algorithm 1 will recover an MAP estimate u^*.

Proof. Note that the hypothesis of the corollary implies that an MAP will never get pruned. Joint with Theorem 3.5.1, we recover the desired result. □

3.6 Implementation Details

The implementation of our algorithm, requires that we are able to compute $P_{\mathcal{U}}(u)$, $P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}$, and p_{thres}. $s_t(T_{switch}|z)$ can be computed via (3.14) and $P_{\mathcal{Z}|\mathcal{U}}(z|u)$ via (3.1). In practice, under the assumptions we make on our device models, $P_{\mathcal{Z}|\mathcal{U}}(z|u)$ is proportional to the error between the observed aggregate result and mean aggregate prediction, thereby simplifying computations. This section describes how each of these objects was instantiated and what further assumptions were made for ease of implementation.

Usage prior

First, we describe our model for the devices to be disaggregated. We modeled each device based on its recorded real power signature from a training dataset. For simplicity, we assumed that each device has only one mode of operation. This restricts the usage pattern for each device at each time step to either signify on or off events, i.e. $u^{(i)} \in \{0, 1\}^{T+1} \subset \mathbb{R}^{T+1}$. We modeled more complex devices with multiple modes as a combination of "elementary" single mode devices. Once disaggregation is achieved, the results for the more complex models can be obtained by aggregating results for their associated elementary devices. More formally, we modeled each device i as $\mathcal{H}^{(i)} : \{0, 1\}^{T+1} \mapsto \mathbb{R}$, and the aggregate system as $\mathcal{H} = \sum_i \mathcal{H}^{(i)}$. $\mathcal{H}^{(i)}(u^{(i)})[t]$ corresponds to the recorded signatures at time t if $u^{(i)}[t] = 1$ and is zero if $u^{(i)}[t] = 0$.

In our implementation, $P_{\mathcal{U}}$ was used as a prior to constrain the duration of usage patterns and computational complexity. We restrict $\mathcal{U}$ to be sampled from the set of inputs so that :

1. Each device is on for at least as long as the minimum amount of it was observed on in the training set.

2. Each device is on for no longer than the duration of it's recorded signature.

3. At most one device changes state at each point in time.

These restrictions implicitly define our prior over the set of u's. Usage patterns where devices are on for longer than the recorded signature duration would force us to run out of data to model the device. With a more sophisticated model, this restriction is unnecessary. We also encourage u to be similar to usage patterns seen in the training set by enforcing that each device must be in use for a minimum amount of time or not at all. Finally, restricting only one state change at each time step improves computational performance. It is equivalent to the assumption that at most one device turns on or off at any time. We have found this assumption to have negligible effects on the results while improving computational speed, if the data is sampled fast enough. Crucially, this restriction also prevents overfitting to noisy aggregate data during test time.

Device model

Next, we instantiated $P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}(y^{(i)}|u^{(i)})$ by assuming that $y^{(i)}$ is distributed normally about the output of the device model given $u^{(i)}$ as an input. Concretely, $P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}(y^{(i)}|u^{(i)}) \sim \mathcal{N}(\mathcal{H}^{(i)}(u^{(i)}), I^{(T+1)\text{x}(T+1)})$.

Thresholds

Extending our notation so that $\mathcal{S}_t(\mathcal{F}|z) = \{s = s_t(T_{switch}|z) : T_{switch} \in \mathcal{F}\}$ and $\mathcal{S}_t(\mathcal{F}|z)_{(k)}$ as the k'th order statistic of $\mathcal{S}_t(\mathcal{F}|z)$, we chose p_{thres} at time t to be :

$$\begin{cases} \mathcal{S}_t(\mathcal{F}|z)_{(N_{\text{max\_filters}})} & \text{if } \alpha\sqrt{t} < \mathcal{S}_t(\mathcal{F}|z)_{(N_{\text{max\_filters}})} \\ \alpha\sqrt{t} & \text{otherwise} \end{cases},$$

where α and $(N_{\text{max\_filters}})$ are hyper-parameters. Effectively, this ensures we keep at least $(N_{\text{max\_filters}})$ filters but no more than however many are $\alpha\sqrt{t}$ close to optimal at time t. In practice, all the assumptions required for optimality may not be fully satisfied. To improve performance in this scenario, we branch not only the most likely filter, but a handful of them. In this implementation, we branch at least 5 filters, but no more than as many as are $\alpha\sqrt{t}$ close to optimal at time t.

3.7 Experimental Evaluations

Our proposed approach was evaluated on the AMPds2 dataset [70]. The AMPds2 dataset contains recorded energy measurement for a residential building in Canada. This dataset contains 11 types of measurement for 21 sub-circuits/devices recorded at one minute intervals over the space of two years. In addition, we made use of NILMTKv2.0 [56] to preprocess the

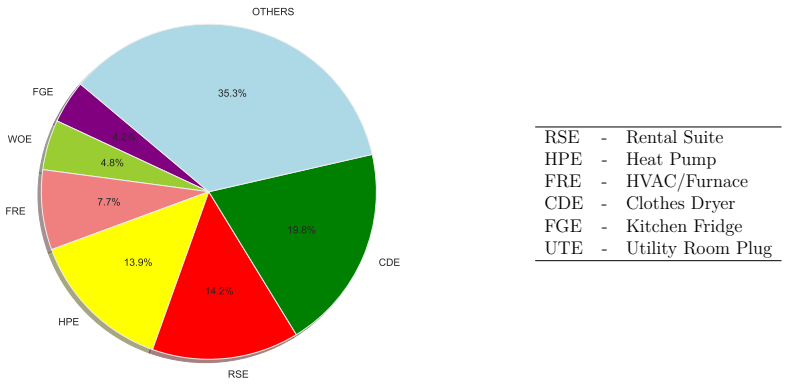

Figure 3.2: Percentage of total energy consumed by each sub-circuit on the AMPds2 dataset.

dataset. This choice was made to allow consistent future comparisons with our work. We compare our work against opensource implementations of two baselines made available via NILMTKv2.0, combinatorial optimization (CO) [43] and Factorial Hidden Markov Model (FHMM).

We made use of the Total Energy Correctly Assigned (TECA) metric to evaluate the goodness of our approach. The TECA metric is a standard metric used in [60]. Denoting the ground truth energy consumption for device i as $y^{(i)}$, its estimate obtained by our algorithm as $\hat{y}^{(i)} = \mathcal{H}^{(i)}(\widehat{u_{\text{MAP}}}^{(i)}[t])$ and $z[t]$ as the aggregate ground truth measurement at time t, Total Energy Correctly Assigned (TECA) :

$$\text{Acc} \ = 1 - \frac{\sum_t \sum_i \left| y^{(i)}[t] - \hat{y}^{(i)}[t] \right|}{2 \sum_t z[t]} \ . \tag{3.16}$$

Since disaggregation errors can be unbounded, TECA scores can take any value in the range $[-\infty, 1]$, $(z[t] \geq 0)$. A good TECA score should be close to 1. We have made use of denoised aggregate values in all our experiments and metrics, meaning that the aggregate measurement is exactly the sum of the measurements obtained from the component parts. Essentially, we are assuming there is no noise introduced by aggregation. The only noise present is that which is introduced during the measurement of the component devices. We also attempted to evaluate the robustness of our approach by including a device not present in the training set as part of the aggregate signal used during test time. We refer to this device as an unmetered appliance.

Furthermore, we only disaggregated a subset of the 21 sub-meters. More specifically, we chose 3 of the top 5 energy consuming devices. Fig. 3.2 shows the percentage energy consumption of these devices. RSE was not initially included as one of the devices comprising the signal to be disaggregated because it is composed of many unknown electrical devices found in a rental unit detached from the main house monitored by the AMPds dataset.

However, we have included it in experiments where we wish to test the robustness of our framework to unmodeled signals, as this realistically represents deployment use cases.

All algorithms tested were given an identical period of two weeks of data as a training set and were tested on an identical set of 3 days of data not included in the training set. Both the training and test sets were selected randomly from the dataset. The training set was used to extract signatures for devices. These signatures were manually extracted from the training set. Parameter $min\_on\_time^{(i)}$, represents the minimum amount of time device i should be "on" for. We estimated its value from the training set by computing the shortest amount of time the device was observed in the "on" state. α was set to 0.01 and

3.8 Results

Results with metered appliances

Fig. 3.3 shows an example estimated power consumption result for our approach and the two other baselines, compared to the ground truth. Table 3.1 shows the results for all three algorithms using TECA as metrics.

The results shown in Fig. 3.3 and Table 3.1 indicate that our algorithm performs comparably to the baselines when all the devices comprising the aggregate signal are known. Our optimization procedure is able to find a sequence of inputs for each device that adequately explains the aggregate signal. We also note that it produces more realistic looking signals.

Results with unmetered appliances

To test for robustness against unmodeled signals that might, for example, be present when not all devices composing the reported aggregate signal are known before hand, or due to signal corruption, we included RSE as part of the aggregate signal, z. Note that this signal was not present in the training set but is present during test time. Fig. 3.4 and Table 3.2 show the disaggregation results in this case.

In this scenario, our algorithm outperforms the baselines. Fig. 3.4 also indicates that it produces more realistic signals as disaggregation estimates for each device. We argue that a key benefit of energy disaggregation is to provide consumer feedback on usage patterns. Unrealistic looking results such as those produced by the baselines (e.g. those for CDE) may lose consumer trust. We also emphasize that the unmetered scenario is the scenario more likely to be encountered during deployment as it can be impractical to obtain a model for every device in the house. Consequently, obtaining realistic estimates in the presence of some unmetered devices is essential for ease of deployment and obtaining consumer trust.

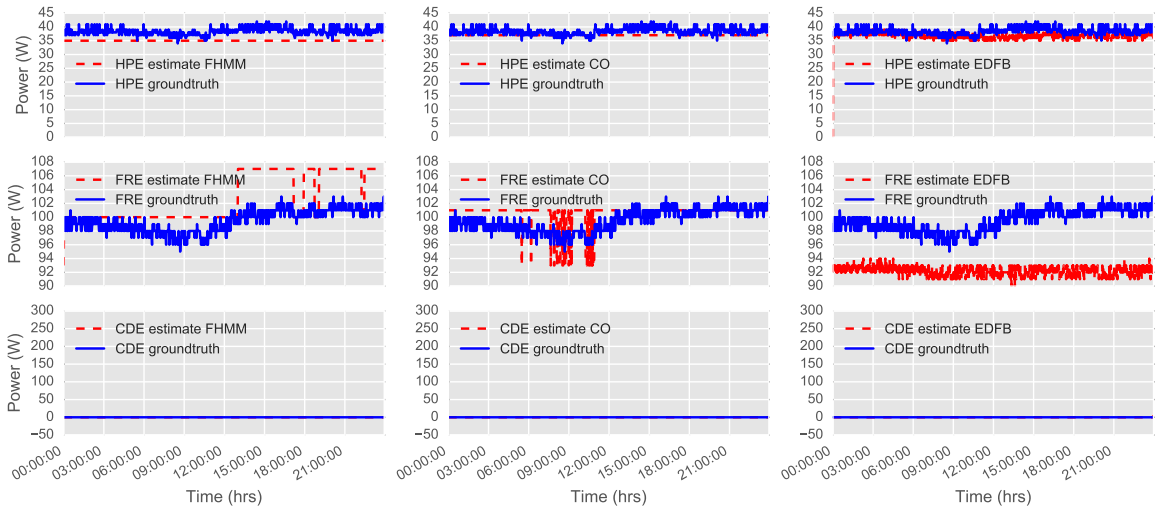

Figure 3.3: Disaggregation output for FHMM , CO and EDFB (ours) on day 1 in the test set. Ground truth also inserted for comparison. Note that our method produces more realistic looking signals, which is important in gaining users' trust of the system.

Table 3.1: Comparison of total energy correctly assigned (TECA) across all three algorithms for each day in the test set. A good TECA score should be close to 1. We achieve a score within 95% of the baselines.

	FHMM	**CO**	**Ours**
Day1	0.975	0.988	0.967
Day2	0.875	0.864	0.871
Day3	0.913	0.904	0.880

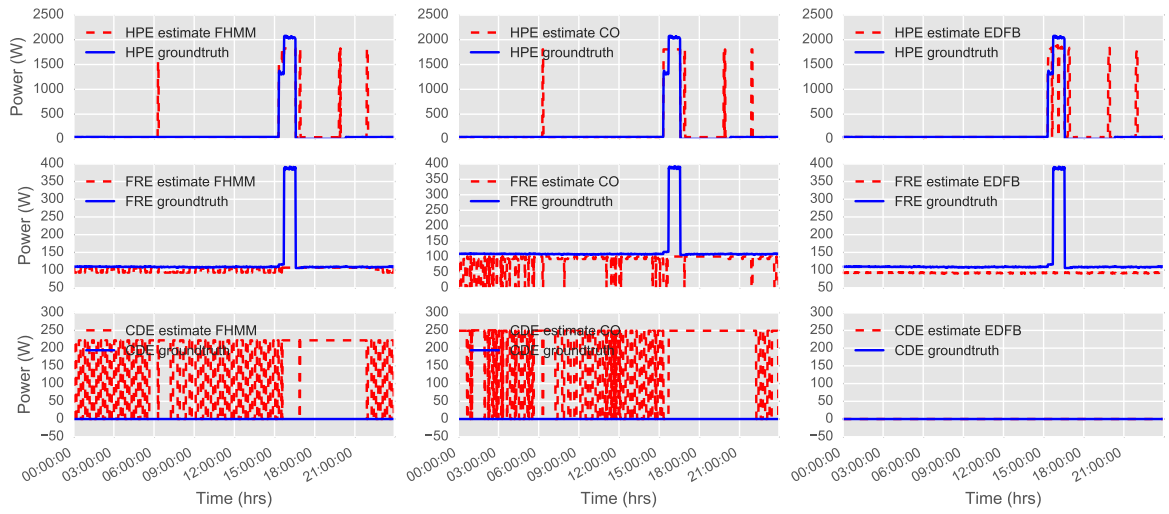

Figure 3.4: Disaggregation output for FHMM, CO and EDFB (ours) on day 3 in the test set after adding a signal not encountered during training. Ground truth also inserted for comparison. Note that our method produces more realistic looking signals, which is important in gaining users' trust of the system.

Table 3.2: Comparison of total energy correctly assigned (TECA) across all three algorithms for each day in the test set after adding a signal not encountered during training. A good TECA score should be close to 1.

	FHMM	**CO**	**Ours**
Day1	−0.187	−0.283	0.342
Day2	0.168	0.641	0.812
Day3	0.592	0.443	0.832

Discussion

We observe that the choice to model devices with recorded signatures and restricting $u^{(i)} \in \{0,1\}^{T+1} \subset \mathbb{R}^{T+1}$ constrains energy consumption predictions to not only take on exactly the same values as those in the recorded signature, but to also transition between these values in exactly the same manner as that observed in the recorded signature for at least as long as $min\_on\_time^{(i)}$ (i.e. Given a starting point in the recorded signature, the sequence of predicted energy consumption values are identical to those of the recorded signature). The other algorithms being compared are not constrained to transition between values in this manner. While this choice of modeling improves the ease of implementation, it is also too restrictive. The authors believe that this is a reason for not matching the ground-truth values better. However, Fig. 3.3 shows that our algorithm finds a very reasonable estimate. With a more sophisticated model for the devices, our algorithm should perform even better. In the unmetered case, we observe a robustness to overfitting by our algorithm. This robustness can be attributed by the ability to incorporate priors and flexible device models in our framework.

In this chapter, we have presented the energy disaggregation problem as a MAP estimation problem with data-driven models for devices (we also provide an alternative neural network based model in the Appendix corresponding to the chapter), and have described an algorithm for solving this problem. We have also shown that our solution is optimal under certain conditions.

In line with the theme of data-driven perception, we note that this algorithm turns raw aggregate data into estimates of appliance usage patterns. Given our data-driven device models, we are able to obtain individual appliance energy consumption estimates (the information we seek) as a function of the usage patterns.

Finally, we have validated our work on a public dataset and highlighted the ability of our framework to provide good solutions more similar in pattern to the ground truth than baselines. At the same time, we achieve more robust score than the baselines on a standard metric. Our approach easily allows the designer incorporate different priors and device models, while still making inference tractable.

Chapter 4

Mixed Reality: Joint Shape Retrieval and Transform Estimation for 3D Scene Reconstructions and Understanding

List of Symbols

S	A 3D shape represented as a mesh.
$\mathbb{S}$	Set of 3D shapes of the same class, and canonical pose and scale, provided by user to the algorithm
$\mathbb{D}$	Dataset of shapes used by the algorithm.
Ω_S	Set of 3D points inside and on the surface of shape S.
$\partial\Omega_S$	Set of 3D points on the surface of shape S.
Φ	Signed distance function.
s, R, t	scale, rotation matrix, translation vector respectively.
$\omega, \psi, \rho, \theta$	Rotation axis, azimuth angle, elevation angle, angle of rotation respectively.

Our final example of data-driven perception algorithms is a perception algorithm for shape and pose estimation from 3D data. On its own, estimating shape and pose is useful for downstream applications such as robotic manipulation, since the shape and orientation of an object informs us on how to manipulate it. An added benefit of this algorithm is that it provides a compact representation for shape and allows for deformation of shapes. This in turn is very useful for 3D reconstruction algorithms that require compact representations as well as robustness to incomplete sensor data. We present this algorithm below as well as preliminary results on its usefulness for 3D reconstruction and representation of indoor scenes geared towards Mixed Reality applications.

4.1 Introduction

The rapid growth of low-cost LIDAR or RGB-D sensors (e.g., Intel RealSense and Apple iPad Pro LIDAR) has led to the growth of applications that require building 3D models of real-world scenes. Such models are useful for virtual immersion and automatic content creation in Augmented and Virtual Reality (AR/VR) where the user experience can be customized to individuals' own spaces. A typical pipeline of the task currently would include Simultaneous Localization and Mapping to generate an SDF representation of the space. Subsequently, mesh or pointcloud models of the scene may be extracted from the SDFs using variants of the Marching Cubes algorithm[67].

While very accurate models can be produced by this pipeline, it is also well known that the models may suffer from missing parts and holes due to occlusion and insufficient scanning coverage of the scene. Furthermore, pointcloud or mesh representations of the scene may not be organized in a way that is intuitive to edit (i.e., they are unordered), and may take up large amounts of memory to store. These characteristics present a significant bottleneck for AR/VR applications to transmit customized user models in real time and to virtually interact with the models.

To alleviate this problem, multiple lines of work such as [110, 48, 66, 10] have suggested that we decompose the scene into objects and their poses and scales. For example, one popular approach is to replace object pointcloud with a suitable Computer Aided Design (CAD) model from a database. The benefit of this is that the model is more likely to be complete and can be stored using its index in the database rather than a full mesh model. However, since models in CAD databases are most likely stored in a canonical pose and scale [17], this approach is only useful when one can estimate the appropriate scale and pose together with recognizing the model category in the database. This leads to a *joint shape retrieval and transform estimation* (JSRTE) problem, which is the focus of this chapter.

Another drawback of the above approach is that given a scene scan, there is no guarantee of finding similar models in the database. Thus we may lose a great deal of representation power if we only rely on referencing the database indexes. Nevertheless, it has been proposed that deep neural network (DNN) based 3D generative models such as [3, 76, 39] allow one to produce a larger set of shapes than are available during training (or in the database) by learning a latent space that allows for interpolation and generalization of shapes. What this implies is that we may now be able to obtain the benefit of using a CAD model representation without sacrificing too much representation power.

In this chapter we utilize an SDF-based 3D DNN model to formulate the JSRTE problem constrained on similarity transforms (rigid body transform and scale) as an optimization problem. We present a gradient-descent based solution and compare our results to the state of the art for shape retrieval and/or transform estimation problems. We show this approach produces excellent results on synthetic and real world data. Our contributions are as follows:

1. Formulate JSRTE problem as a novel joint optimization problem.

2. Demonstrate a parameterization of the problem allowing for easy incorporation in popular DNN frameworks.

3. Conduct experiments on synthetic and real datasets showing the effectiveness of our approach on JSRTE problem and as a form of 3D data compression.

4. Provide extensions to the main algorithm to incorporate prior knowledge such as symmetry and repetition of objects.

5. Provide preliminary qualitative results for reconstructing indoor 3D scenes.

4.2 Related Work

3D Shape Alignment using Signed Distance Functions

Aligning 3D shapes is a well studied problem with varying solutions depending on the representations of the 3D shape. For example, when the geometry of a 3D object is represented using pointcloud, solutions for estimating rigid-body and orientation-preserving similarity transformations have been proposed [44, 45, 8, 114, 111].

Alternatively, and more pertinent to this chapter, 3D shapes can be aligned using signed distance functions (SDFs). [97] provided a good overview of some of these methods. For example, [94, 16] presented techniques to estimate rigid-body transformations between two shapes represented using SDFs. [69, 21] further tackled the problem of estimating rigid-body transformation and scale using SDFs, the same type of transformation studied in this chapter. Their algorithms made use of geometric moments and the Fourier transform. In comparison, we develop a gradient-descent based algorithm to find a solution to an optimization formulation. Similarly, [75, 47] also made use of gradient-descent algorithms on optimization problems to estimate similarity transformations between SDFs. Yet, none of the works above explicitly allow for solving shape retrieval and transformation estimation in one optimization problem. Rather, they only deal with transformation estimation given the shape model.

3D Features for Shape Retrieval and Alignment

There exists a body of work that explicitly deals with shape retrieval. Classical shape retrieval algorithms such as [90, 42, 107] usually follow a two-step approach: The first step involves shape recognition from a database by making use of shape descriptors such as [108, 86, 5]. The second step involves a correspondence search using 3D keypoints and descriptors followed by registration. It has been argued in [93] that dense alignment via SDFs can produce much better results since it does not make use of a correspondence search which may introduce errors. Our work favors this approach. We will compare our superior results against classical 3D feature based retrieval techniques.

More recent work has shifted from using hand-crafted descriptors to data driven ones such as [117, 23]. Alternative data driven approaches [80] make use of deep neural networks (DNNs) to replace the recognition module entirely. However, the sequential approach of recognition and then registration does not allow for corrections of errors in the registration phase caused by a mismatch during the recognition stage. To address this problem, [9] proposed an end-to-end shape retrieval and alignment module. In comparison, although our work also jointly optimizes over the space of shapes and transforms, the formulation is not a feed-forward neural network but an optimization problem that makes use of a neural network model. The main benefit of this is that the optimization process can be viewed as introducing feedback that allows iterative minimization of errors from predictions made by the network. A feed-forward module can only make a prediction once per input and has no mechanism to minimize errors after training. More importantly, the space of shapes over which we optimize allows us to retrieve objects that were not seen during training.

4.3 DeepSDF-based Shape Retrieval and Similarity Transform Estimation

Problem Statement

Assume that we are given a set of shapes $\mathbb{S} = \{S_1, S_2, S_3, \ldots, S_n\}$ belonging to the same class, all in the canonical scale and pose (e.g., a set of chairs all normalized to lie in the unit sphere with forward directions aligned with the positive y axis). Further assume the shapes are represented as meshes although our approach also works for other representations as long as we can extract a signed distance function from the representation.

Then, for a new shape S belonging to the same class but not necessarily in $\mathbb{S}$ or in the same canonical scale and pose, we would like to firstly build a dataset $\mathbb{D}$ of shapes using $\mathbb{S}$, and secondly find an item in $\mathbb{D}$ and a corresponding similarity transformation that best approximates S.

Note that we do not enforce $\mathbb{D} = \mathbb{S}$, but only require that $\mathbb{D}$ is constructed using only $\mathbb{S}$. Typical shape retrieval problems tend to set $\mathbb{D} = \mathbb{S}$. However, as we will show, this may restrict the richness of $\mathbb{D}$. Our approach assumes that all given shapes have an associated SDF. This can be easily obtained, for example, from a triangle mesh representation of the shape.[1]

As shown in Fig. 4.1, the input to our algorithm is a query shape S and a set of canonical shapes $\mathbb{S}$. Its output are the optimal shape representation in canonical pose and scale from $\mathbb{D}$ as a mesh, the optimal similarity transformation, and a compact representation of the query shape for data compression.

[1]Please refer to the appendix for more details on SDF extraction and sampling.

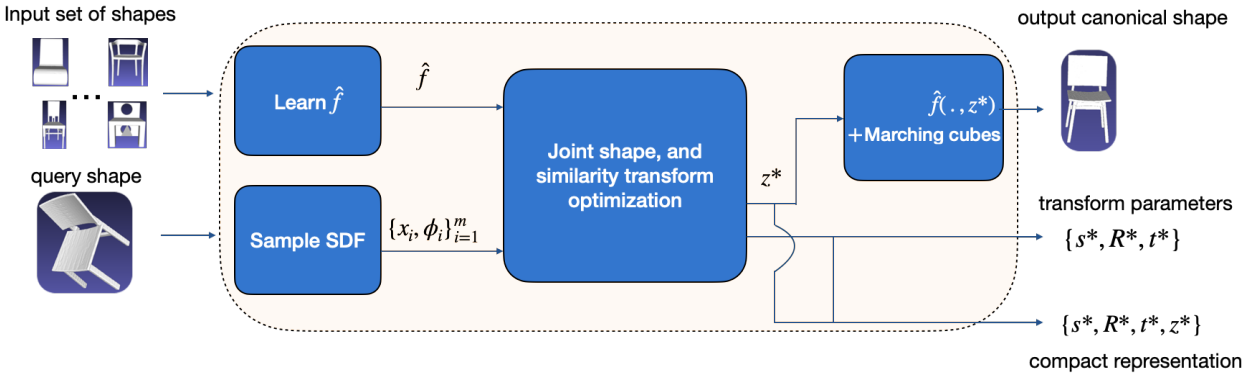

Figure 4.1: Overview of the proposed JSRTE algorithm.

Notation and Preliminaries

A 3D similarity transformation $g \in Sim(3)$ is a tuple (s, R, t), with $s \in \mathbb{R}^+$, $R \in SO(3) \subset \mathbb{R}^{3\times3}$, $t \in \mathbb{R}^3$. Then, g acting on a point $x \in \mathbb{R}^3$ is given by

$$g(x) = sRx + t. \tag{4.1}$$

Let S be a solid 3D shape with a closed surface such that we have a well-defined notion of inside and outside. In addition, let $\partial\Omega_S$ represent the set of points on the surface of S, Ω_S represent the set of points on the surface and interior of S, and Ω_S^c the complement of Ω_S. Then an SDF is an implicit representation of the shape defined as:

$$\Phi(x, \Omega_S) = \begin{cases} -\min\limits_{y\in\partial\Omega_S} ||x-y||_2 \ , \text{ if } x \in \Omega_S; \\ \min\limits_{y\in\partial\Omega_S} ||x-y||_2 \ , \text{ if } x \in \Omega_S^c. \end{cases}$$

Intuitively, for a fixed shape S, its SDF at a point x tells us the distance from x to the surface of S, with the sign determined by whether x lies in the interior of S.

Let S, S' be two shapes related by a similarity transformation g as defined in (4.1), i.e., $\partial\Omega_{S'} = \{g(x) | x \in \partial\Omega_S\}$. Under this condition, it is well known [75] that

$$\Phi(g(x), \Omega_{S'}) = s\Phi(x, \Omega_S). \tag{4.2}$$

In other words, SDFs are invariant to rotations and translations, but vary proportionally with isotropic scaling. The constant of proportionality is the scaling factor s.

Approach

Our proposed approach first learns a dataset $\mathbb{D}$ from the input set of shapes $\mathbb{S}$ and then formulates and solves an optimization problem to jointly estimate the transformation and

shape parameters. Inspired by [76], we also learn a generative model and latent space of shapes. The benefit to this is that the latent space is more expressive than the input set of shapes $\mathbb{S}$. Specifically, the latent space has been shown in [76] to extrapolate the shapes in $\mathbb{S}$ and produce new shapes of the same class but not contained in $\mathbb{S}$. The generative model also allows us to formulate searching over the latent space as a tractable optimization problem. In addition, since the dimension of the latent space is relatively small and a similarity transformation involves only a few parameters, we can simultaneously obtain a compact representation of the query shape.

Approximating SDFs with DNNs and Learning a Latent Dataset

Given an object class of shapes in a canonical pose and scale, $\mathbb{S} = \{S_1, S_2, S_3, \ldots, S_n\}$, one can learn a generative model $\hat{f}(x, z) \approx \Phi(x, h(z))$ using DNNs [76]. Here $x \in \mathbb{R}^3$ is a point and $z \in \mathbb{R}^d$ is a latent vector with one-to-one correspondence with the shapes in $\mathbb{S}$. $h(z)$ is a mapping that assigns each latent vector in $\mathbb{R}^d$ to a shape in $\mathbb{S}$, i.e., $h : \mathbb{R}^d \to \mathcal{P}(\mathbb{R}^3)$ such that $\forall S \in \mathbb{S}\ \exists z \in \mathbb{R}^d : h(z) = \Omega_S$. The dimension d is a design choice, and we refer to $\mathbb{R}^d$ as the latent space. It was also shown in [76] that in learning $\hat{f}$, one learns a latent space that may generate new shapes not in the training dataset.

The implication of these results is that we can choose the latent space as the dataset for our algorithm. Also, given a shape S in the same canonical scale and pose and a set of m samples of its SDF $\chi_S = \{(x_i, \phi_i = \Phi(x_i, \Omega_S)\}_{i=1}^m$, S can be estimated by solving the following problem [76]:

$$\min_{z \in \mathbb{R}^d} \sum_{(x_i, \phi_i) \in \chi_S} |\hat{f}(x_i, z) - \phi_i|. \tag{4.3}$$

The above problem can be viewed as a special case of shape retrieval, since one can convert the latent vector back into a mesh by using $\hat{f}$ to generate SDF samples of the shape and using [67] to convert it to a mesh. Moreover, in the event that we have incomplete knowledge of the SDF of S, this formulation gives some robustness and has been shown to be useful for shape completion as well.

However, the above restriction to shapes in the canonical pose and scale limits the usefulness of this work in real world applications, since most objects are not in a canonical scale and pose. While it is possible to learn a new generative model on a larger dataset containing scaled and transformed versions of shapes in $\mathbb{S}$, we find that this is an inefficient approach to solving the problem. Our main contribution in this work is to present a principled way to overcome this problem without making use of any more data or computation for training.

Joint Optimization of Shape and Transform Parameters

Our main idea is to modify (4.3) to include an optimization over similarity transform parameters that is also easy to implement. Using (4.2) and (4.3), we formulate the problem

as:

$$\min_{s,R,t,z} \sum_{(x_i,\phi_i)\in\chi_S} \left| s\hat{f}\left(\frac{R^{-1}(x_i-t)}{s}, z\right) - \phi_i \right| \tag{4.4}$$

with $s \in \mathbb{R}^+, R \in SO(3), t \in \mathbb{R}^3, z \in \mathbb{R}^d$. Then one may use gradient-descent algorithms to solve the problem. However, care must be taken since R does not live in a vector space. While it is possible to properly take gradient steps in $SO(3)$ [22], we take an alternative approach to parameterize R.

Using the axis-angle representation for rotation matrices, we know that for any rotation matrix R, there exists $\omega : ||\omega||_2 = 1$ and $\theta \in [-\pi, \pi] : R = \exp(\hat{\omega}\theta)$, where $\omega = [\omega_1, \omega_2, \omega_3]^\top$, $\hat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$. Since ω lies on the unit sphere, we may parameterize it using spherical coordinates as $\omega_1 = sin(\psi)cos(\rho),\ \omega_2 = sin(\psi)sin(\rho),\ \omega_3 = cos(\psi)$, where, to avoid potential confusion due to overloading symbols, we have used ψ to represent the polar angle and ρ to represent the azimuthal angle. Consequently, we may write

$$R(\psi,\rho,\theta) = \exp\left(\begin{bmatrix} 0 & -cos(\psi) & sin(\psi)sin(\rho) \\ cos(\psi) & 0 & -sin(\psi)cos(\rho) \\ -sin(\psi)sin(\rho) & sin(\psi)cos(\rho) & 0 \end{bmatrix}\theta\right) \tag{4.5}$$

and the initial problem (4.4) as

$$\min_{\substack{s,\psi,\rho,\\ \theta,t,z}} \sum_{(x_i,\phi_i)\in\chi_S} \left| s\hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(x_i-t)}{s}, z\right) - \phi_i \right|. \tag{4.6}$$

In this way, no special handling of gradients has to be taken. We find that this explicit formulation allows for easy implementation and incorporation into common deep learning frameworks with automatic differentiation such as [78]. We summarize our JSRTE algorithm in Algorithm 2.

Implementation

We have implemented JSRTE algorithm using PyTorch [78] and report some implementation details here for reproducibility. Specifically, the matrix exponential $\exp(\hat{\omega}\theta)$ is expressed by Rodrigues' formula [85, 68]: $\exp(\hat{\omega}\theta) = I + \hat{\omega}sin(\theta) + \hat{\omega}^2(1 - cos(\theta))$. Adam[58] as implemented in [78] is adopted to solve the optimization problem (4.6). To ensure that the search is in the feasible set of the problem, $s \in [\underline{s}, \bar{s}]$ can be restricted as $\underline{s} = 0.01, \bar{s} = 10$. $\hat{f}$ is learned using open source code provided by [76, 33], with latent vectors in $\mathbb{R}^{256}$. For all objects, we have trained on subsets of the corresponding Shapenet [17] class. To encourage convergence of the network, [76] has used a penalty term on the norm of z weighed by 10^{-4}. We also include this term in solving (4.6). We generally make use of the same SDF extraction and sampling approach used in [76] (see appendix for details).

Algorithm 2 Joint Shape Retrieval and Transform Estimation (JSRTE)

function JSRTE($\mathbb{S}, S, s^0, \rho^0, \psi^0, \theta^0, t^0, z^0$)
  $\hat{f}, \mathbb{D} \leftarrow \mathbb{S}$ ▷ Learn generative model and latent space from data (offline).
  $\{x_i, \phi_i\}_{i=1}^m \leftarrow S$ ▷ Sample SDF from query shape
  Initialization ▷ Initialize (4.6) using provided parameters
  Solve (4.6) for s^*, R^*, t^*, z^* ▷ Use gradient descent to solve (4.6)
  $\{\bar{x}_i, \hat{f}(\bar{x}_i, z^*)\}_{i=1}^n \leftarrow \hat{f}, z^*, \bar{x}_i \in [-1, 1]^3$ ▷ Generate SDF samples at z^* using $\hat{f}$
  $S^* \leftarrow$ MARCHINGCUBES($\{\bar{x}_i, \hat{f}(\bar{x}_i, z^*)\}_{i=1}^n$) ▷ Generate mesh from SDF samples with [67]
  return s^*, R^*, t^*, z^*, S^*

4.4 Experiments

We perform three experiments to validate our approach for shape and transformation estimation. The first experiment seeks to evaluate the performance of the proposed algorithm when we initialize its parameters within reasonable bounds of the groundtruth. The second experiment evaluates our algorithm on real world data and compares it to state-of-the-art benchmarks. The third experiment assesses the viability of our approach as a means for 3D data compression.

Synthetic data

Gradient-descent based methods are susceptible to converging to locally optimal solutions. So it is important to find good initialization points for the variables. For this experiment, we want to observe the behavior of our algorithm when initialized within reasonable bounds of the groundtruth.

We sample objects from the chair, table, sofa, and bed categories from ShapeNet [17]. First, 30 objects from each category are sampled. Then for each object a random scale and pose are assigned by sampling the groundtruth parameters, $s \sim \text{Uniform}([\frac{1}{2}, 2))$, $\psi, \rho, \theta, \sim \text{Uniform}([-\pi, \pi))$, and $||t||_2 \sim \text{Uniform}([0, 4))$. The direction of t is chosen uniformly from the surface of a unit sphere. For each shape and transform pair, we solve the shape retrieval problem using our algorithm 50 times, with each trial using a random initialization. We carry out two sub-experiments to highlight scenarios where our algorithm is applicable. In the first scenario, we assume that we know the axis of rotation (and consequently ψ, ρ), but all other parameters are unknown. This is a reasonable assumption since many objects lie upright on flat surfaces such as floors or tables. One may use the normal to the surface as the axis of rotation. In the second scenario, we relax this assumption and introduce some uncertainty in the axis of rotation. The scenarios are used to randomly sample the initialization for our algorithm.

Concretely, we refer to the initialization parameters as $s^0 = s(1 + \Delta s), \psi^0 = \psi + \Delta\psi, \rho^0 = \rho + \Delta\rho, \theta^0 = \theta + \Delta\theta, t^0 = t + \Delta t$. The Δ variables correspond to the difference between the ground-truth and initialization parameters. We then sample the Δ variables uniformly

from ranges determined by each scenario. Table 4.1 below summarizes these ranges for the different scenarios.

Table 4.1: Parameter ranges for the difference between initialization points and groundtruth values.

Scenario	Δs	$\Delta\psi$	$\Delta\rho$	$\Delta\theta$	$\|\|\Delta t\|\|(m)$
Known rotation axis	$[0, 0.3)$	N/A	N/A	$[-\frac{\pi}{9}, \frac{\pi}{9})$	$[0, 0.15)$
Unknown rotation axis	$[0, 0.3)$	$[-\frac{\pi}{36}, \frac{\pi}{36})$	$[-\frac{\pi}{36}, \frac{\pi}{36})$	$[-\frac{\pi}{9}, \frac{\pi}{9})$	$[0, 0.15)$

In all scenarios, We choose the direction of Δt uniformly from the surface of a unit sphere. $z^0 \sim \text{Normal}(\mathbf{0}, \sigma\mathbf{I}), \sigma = 0.01$. We use the corresponding ShapeNet [17] model class as input shape set $\mathbb{S}$.

To quantitatively measure the performance of our algorithm, we use the F-score between the output shape (transformed by our predicted transform parameters) and the test shape as recommended in [101]. The F-score measures the similarity between 3D surfaces as the harmonic mean between precision and recall. A point on the predicted surface is considered a true positive if it is within a threshold ϵ of the groundtruth surface. In this work, we have set this threshold as 5% of the scale of the groundtruth object. We refer to this score as F@5%. The F-score ranges from 0 to 1, with higher numbers indicating more closely matched shapes.

For each shape and transform pair, we categorize the F-score into bins and count the number of initializations out of 50 that fall into each bin. Then for each bin, we report aggregate statistics using box plots in Fig. 4.2. The position of each box-plot on the x-axis corresponds to the rightmost edge of its bin. The leftmost edge of its bin is the previous point on the x-axis.

Fig. 4.2 corresponds to scenarios with a known and unknown axis of rotation respectively. In both scenarios, we observe that our algorithm performs well with most F@5% scores greater than 0.8. We also observe a slight degradation in performance in the scenario with an unknown axis of rotation, as is to be expected due to greater uncertainty. Intuitively, the results imply that with a known axis of rotation, if we know the amount of rotation up to $\pm 20°$, the translation up to ± 15 cm and the scale up to 30% of the true scale, we can expect our algorithm to perform very well. We also perform more experiments to test the our algorithm in other parameter ranges and report those as well as qualitative results in the supplementary.

Real Data

In the next set of experiments, we evaluate our algorithm on real data. In this chapter, we demonstrate the results on the chair subset of the Redwood dataset [19]. The Redwood dataset contains scans (represented as triangle meshes) of real objects in varying scales and poses. We have manually segmented the dataset so that we remove other objects in the

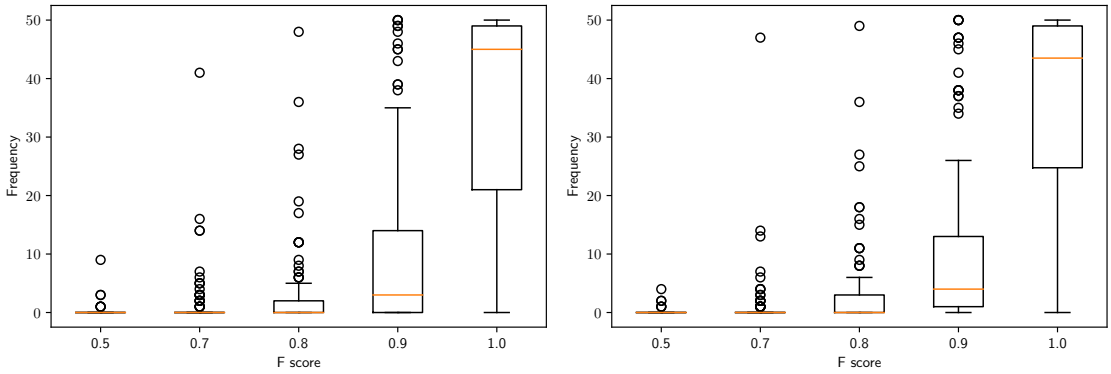

Figure 4.2: Box plot of F@5% from experiments described in Table 4.1. **Left:** Scenario with a known axis of rotation; **Right:** Scenario with an unknown axis of rotation. The median for each bin is shown as an orange line in the box, with outliers shown as circles. Each box is placed on the right edge of its corresponding bin, the left edge of each bin is the right edge of the preceding bin.

background, leaving behind only the floor and the main object in each scene. We assume that all objects in the scene lie on the floor and are upright. This allows us to obtain an estimate for ω as the normal to the floor plane. In this way, we collect 89 shapes and samples of their SDFs. However, we use 31 randomly selected shapes from these 89 for validating the baselines described below, leaving only 58 for testing. We discarded all other scenes that could not be segmented properly. For each shape, we initialize s using the radius of its bounding sphere and t using the center of the bounding sphere. We estimate ω using the normal vector to the floor plane and z is initialized by drawing from $\text{Normal}(\mathbf{0}, \sigma\mathbf{I}), \sigma = 0.01$. The only parameter left is θ. For θ we grid up the space in increments of 30° and run our algorithm for each value of $\theta^0 \in [0, \frac{\pi}{6}, \frac{\pi}{12}, \ldots 2\pi)$. We then select the best result from each of these initialization of θ using the F@5% score.

We compare our work to three different approaches for shape retrieval and transform estimation. In these approaches, we perform recognition to find the closest shape to the query shape in the set of input shapes and then perform registration to estimate a similarity transform.[2]

The first approach uses Harris3D keypoints and the SHOT[89] descriptor to perform recognition and provide correspondences for registration. Registration is then carried out using [111]. Specifically, for recognition, we describe each shape in the dataset using the descriptor and keypoint detecting method stated above. During test time, we compare the keypoints and descriptors of the query shape to those in the dataset. This method is referred to as Harris3D+SHOT.

The second approach replaces the keypoint detector and descriptor with the shape

[2]We are aware of other state-of-the-art SDF based methods for jointly estimating shape and similarity transformation [9]. However, we could not find a publicly available implementation to use for testing.

descriptor developed in [5]. For registration, points are sampled from the mesh and then applied to an Iterative Closest Point (ICP)[14] variant that makes use of [111] for transform estimation. Correspondences are assigned by solving an optimal assignment problem. We find that this significantly improves the result. This method is referred to as OURCVFH+ICP.

The last approach uses PointNet[81] as described in [80] for recognition and the ICP method described above for registration. This method is referred to as PointNet+ICP.

We report the average F@5% score of these methods in Table 4.2, which shows our algorithm significantly outperforms the three alternative approaches. We also provide qualitative results of randomly sampled test cases in Fig. 4.3, which shows that the shapes predicted by our method are qualitatively better than the alternative methods, with much less misalignment. Alternative methods using a two-step pipeline of recognition and registration can propagate errors in the recognition stage that cannot be corrected by just a similarity transformation in the registration stage. In addition, our method uses a richer search space for shapes by using the latent space as its dataset, rather than just the set of input shapes provided. Moreover, finding good correspondences can be difficult for other two-step approaches as clean CAD models and noisy real scans have differing local surface properties. Finally, the need to sample the surface to allow tractable registration may introduce errors in the sense that true corresponding pairs may not belong to the sampled sets. Our method on the other hand does not make explicit use of correspondences.

Table 4.2: Quantitative results on Redwood dataset. Average F-scores for shape retrieval and similarity transformation methods compared to ours. Our algorithm's score more than doubles the others.

	Harris3D + SHOT	OURCVFH + ICP	PointNet + ICP	Ours
F@5%	0.273	0.367	0.452	**0.902**

3D Shape Compression

Finally, we conduct an experiment to explore the viability of our approach as a form of 3D data compression. Given the neural network parameters of $\hat{f}$, we can choose to store only the latent vector representing a shape and the associated similarity transform. With access to $\hat{f}$ and the transform parameters, one can then decode the latent vector and convert it back to a mesh model. In our experiment, the network size is approximately 7MB. This is a constant cost shared across many uses and is consequently amortized.

For this experiment, we save each mesh in our Redwood test set under two mesh simplification schemes, mesh simplification using Quadric Error Metrics[37, 119] and Vertex Clustering [119]. We have varied the parameters of each method. For mesh simplification using Quadric Error Metrics, we set the target number of faces to be reduced by a factor of 100 and 1000 from the number of faces in the original mesh. We refer to these experiments as QE-100, QE-1000 respectively. For Vertex Clustering, we vary the size of the grid cells

used for clustering to be a factor of 0.1 and 0.2 of the radius of the bounding sphere for the shape. We refer to these as VC-0.1, VC-0.2 respectively. We save all meshes and compute the average size of the files as well as the mean F-score (compared to the original mesh) to give an indication of the representation power at each simplification parameter. We compare these scores against our mean F-score and the cost of saving the latent vector and transformation parameters as a 4×4 matrix and report the results in Table 4.3. The result shows that our method provides an impressive tradeoff between reconstruction quality and storage size.

Table 4.3: Comparison of storage size and representation power for various mesh simplification algorithms. Our approach provides an excellent trade-off between accuracy and storage space, saving over 20x space as the most accurate method with less than a 10% drop in accuracy.

	Ours	QE-100	QE-1000	VC-0.1	VC-0.2
Average F-score	0.902	0.977	0.778	**0.978**	0.839
Average size (KB)	**1.088**	20.120	3.454	34.833	9.073

Extensions

While we have presented the main idea and results for Joint Shape Retrieval and Transformation Estimation in previous sections, we now take a look at special scenarios where we may be able to take advantage of prior knowledge to obtain more robust results.

Reflective Symmetry

In 3D space, a reflection is a Euclidean (distance preserving) transformation with a plane as its set of fixed point. We say a set of points Ω is symmteric with respect to the transformation g if $g(\Omega) = \Omega$. Man-made objects typically enjoy reflective symmetry. For example most chairs and tables exhibit bi-lateral symmetry.

Symmetry has been used in previous 3D reconstruction work [96, 73, 116, 53] as a way to obtain more robust reconstruction or recognize shapes. Indeed with known reflective symmetry, partial observations of an object may be as good as complete knowledge of the object. Consequently, knowing that an object has one or multiple reflective symmetries allows us to obtain robust shape retrieval and transformation estimation. To take advantage of this knowledge in our framework, first assume the canonical shape has reflective symmetry with respect to the transform g (not to be confused with notation given earlier for similarity

transforms). Then we may enforce this symmetry in our optimization problem as[3]:

$$\min_{\substack{s,\psi,\rho,\\ \theta,t,z}} \sum_{(x_i,\phi_i)\in\chi_S} \left| s\hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(x_i-t)}{s}, z\right) - \phi_i \right| + \lambda \left| \hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(x_i-t)}{s}, z\right) - \hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(g(x_i)-t)}{s}, z\right) \right|. \quad (4.7)$$

The equation (4.7) approximately enforces symmetry by making use of the prior information on symmetry as a penalty term with λ as a parameter controlling the importance of the penalty. We note that the penalty term

$$\left| \hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(x_i-t)}{s}, z\right) - \hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(g(x_i)-t)}{s}, z\right) \right|.$$

is obtained by approximating the relation:

$$\Phi(g(x), \Omega_{S'}) = \Phi(x, \Omega_S) \ \forall x \in \mathbb{R}^3.$$

Since Φ is a scalar function, we can show the above relation to be true if we show that:

1. $\text{sgn}(\Phi(x, \Omega_S)) = \text{sgn}(\Phi(g(x), \Omega_{S'}))$.
2. $|\Phi(x, \Omega_S)| = |\Phi(g(x), \Omega_{S'})|$.

We note that since $\text{sgn}(\Phi(x, \Omega_S))$ is determined by whether or not $x \in \Omega_S$, the sign is preserved if g is a function that maps interior (boundary/ points outside the shape) points in the domain to interior (boundary/ points outside the shape) points in the image and is also injective. The magnitude of $\Phi(x, \Omega_S)$ is determined by the distance of x to points on the boundary of Ω_S, thus the magnitude is preserved if g is a distance preserving function. Both statements are shown to be true by making use of Lemmas B.2.1, B.2.2, B.2.3 and the fact that g is a Euclidean transform.

In Fig 4.4, we provide qualitative results of this extension. We have manually detected planar symmetries in randomly sampled shapes from the ShapeNet dataset [17] and simulated scenarios with missing data and occlusions. We use this occluded data as input to our algorithm, including information about reflective symmetries. We compare the results to an approach that does not use this knowledge.

As can be seen in Fig. 4.4, including reflective symmetries can provide more accurate and aesthetically pleasing results even in the presence of severe occlusions.

[3]see Appendix for full derivation.

Repeated Objects

In addition, many indoor environments include multiple instances of the same shape. For example, a dinning room might contain four identical chairs around a dining table or a classroom might contain multiple identical desks. Knowing that multiple instances of the same shape occur in a scene can be very useful in improving the robustness of results obtained by our proposed algorithm.

Specifically, multiple instance of the same object provide multiple views of the object. This is especially useful since occlusion patterns are generally not the same across the object (the legs of one chair may be occluded but the legs of another instance of the chair may not be). When all the information from all the instances of the object are used together, we obtain a better picture of the shape in question. Furthermore, if we know that all instances are of the same scale, we can also include this as a constraint in our optimization.

With this in mind, for K instances of the same shape, our optimization problem becomes:

$$\min_{\substack{s^k,\psi^k,\rho^k,\\ \theta^k,t^k,z\\ k=1,\ldots,K}} \sum_{k=1}^{K} \sum_{(x_i^k,\phi_i^k)\in\chi_S^k} \left| s^k \hat{f}\left(\frac{[R(\psi^k,\rho^k,\theta^k)]^{-1}(x_i^k - t^k)}{s^k}, z \right) - \phi_i^k \right|. \tag{4.8}$$

where the superscripts have been used to indicate parameters and data belonging to a particular instance.

We have tested this formulation by randomly sampling shapes from the ShapeNet dataset [17] and created three instances of the shape at different poses. We solve the JSRTE problem making use of the knowledge that the shapes are multiple instances of the same shape. We also solve the JSRTE problem without making use of this information. Each triple of rows in Figs. 4.5, 4.6 and 4.7 show the results of this experiment for a sample shape. As can be seen in the figures, incorporating knowledge of multiple instances of the same shape provides significant benefits in obtaining good results with our algorithm. In particular, we observe that even when one of the instances undergoes severe occlusions, we may still be able to obtain excellent results by making use of information from the other instances.

Layout Modeling

We stated earlier that an indoor scene may be the decomposed into a layout as well objects and their poses and scales. Previous sections have dealt with how to represent and estimate objects, we now focus on layout estimation and representation. Indeed, the distinction between objects and layouts with respect to the algorithm presented in this chapter is rather artificial since SDFs may be used to represents shapes irrespective of their class. As such,we may be able to respresent layouts using deep generative models. To test this hypothesis, we have collected a set of 3D CAD models from the SUNCG dataset[95] as well as publicly available models from the 3D Warehouse[1]. We train DeepSDF [76] on 2175 model and display some sample outputs generated by the model in Figs. 4.8 and 4.9.

Figures 4.8 and 4.9 validates our hypothesis. We also note that we are able to represent both planar and non-planar surfaces with the same amount of data. With this representation, we do not need to make an assumption that the layout of the scene is planar or Manhattan, giving us more representation power and flexibility in the type of scenes we can represent.

To test the ability of our proposition to model indoor scenes, we have obtained two scans of indoor rooms from the Replica 3D dataset [98] and used the proposed methods to model the layout and objects. Figures 4.10 and 4.11 below show some of the qualitative results displaying very promising potential.

4.5 Interpretation as an Estimation Problem

The JSRTE problem presented in this chapter is an estimation problem. Concretely we seek to estimate from observed SDF data the following parameters z, s, ψ, ρ, θ and t corresponding to shape and similarity transform parameters.

To obtain the likelihood function, we model our observations at each point x_i as a function of the data-driven generative model corrupted by noise. We also assume that the noise variable at each point is independent. Specifically, we model our observations as:

$$\phi_i = s\hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(x_i - t)}{s}, z\right) + W_i$$

for $i = 1, \ldots, m$, where W_i is a random variable representing the noise corruption and the W_i's are independent. Our presentation in this chapter assumes that the noise variables follow a Laplace$(0, 1)$ distribution. This assumption was made because it empirically produced better results. Intuitively, this makes sense since the Laplace distribution has "fatter" tails than the Gaussian distribution and thus permits the observation of more noisy data.

The optimization problem in (4.6), makes use of an uninformative prior over the variables of interest, however as mentioned in section 4.3, it may be beneficial to assume that the latent vector z is close to zero and within the unit sphere. In addition, including information about repeated objects and symmetry also induce a prior on the latent variables. This is also shown to produce significant improvements and robustness.

The algorithm presented in this chapter is essentially an algorithm that seeks to find the Maximum A Posteriori (MAP) estimate given the likelihood and prior functions. It is important to note that we only provide an approximation to the MAP estimate since we make use of gradient descent solvers that may be stuck at locally optimal points.

4.6 Discussion

In this chapter we have presented a formulation for shape and similarity transform estimation from SDFs as an optimization problem. We have obtained good results with a gradient-descent optimization scheme and good initialization of the parameters. We have also shown results on a dataset of real scans and obtained superior performance to benchmarks.

From the perspective of perception, we present an algorithm to convert raw depth measurement (as contained in a SDF) and object classification into information about the pose of an object and its identity with respect to a fixed dataset (constructed latent space). This information is useful as a form of data compression as well as for other tasks such as 3D reconstruction and object manipulation.

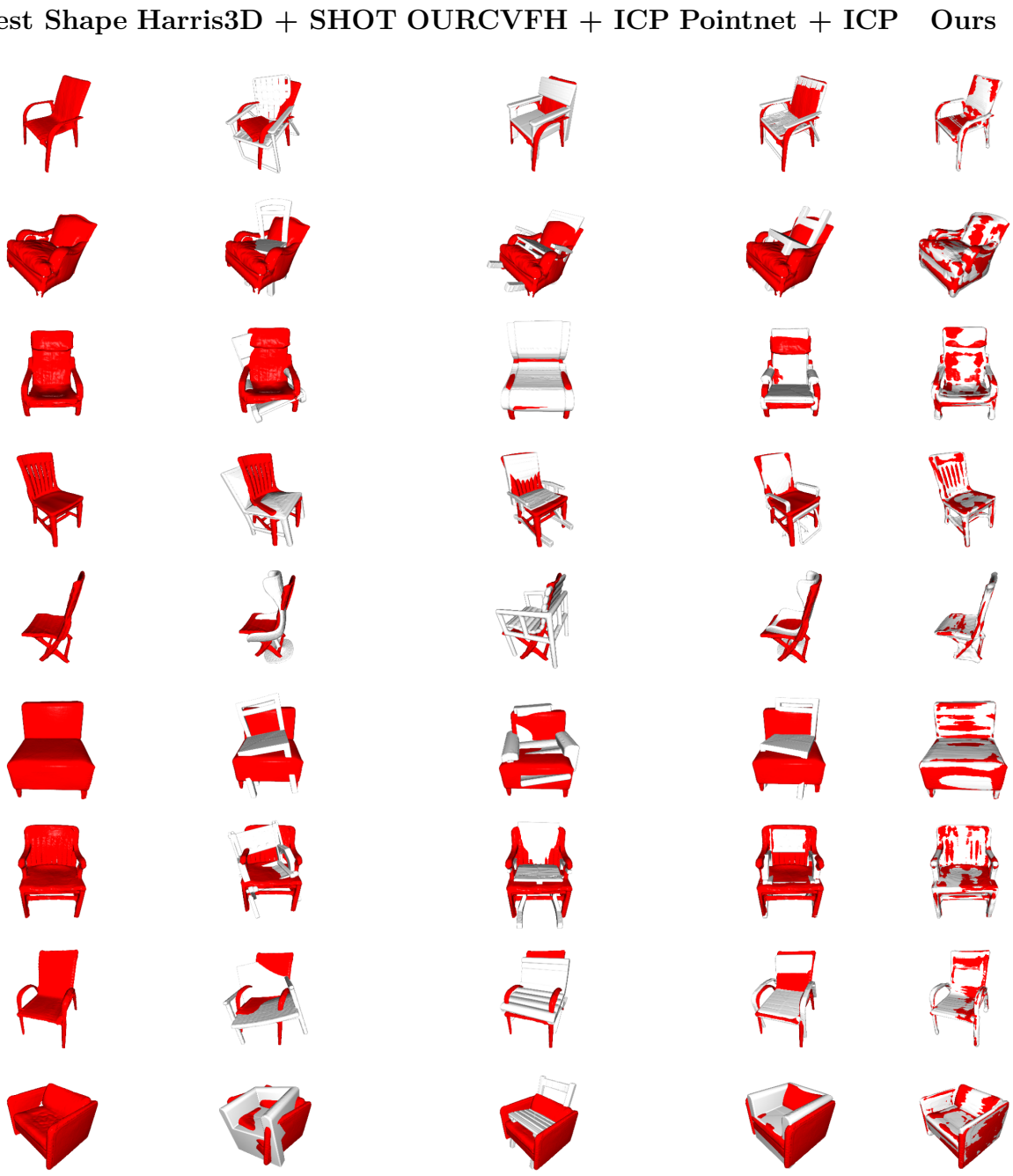

Figure 4.3: Qualitative result on Redwood dataset. Left to Right, test shape, results using Harris3D and SHOT[89] for shape retrieval, results using OURCVFH [5] and ICP, results using PointNet [81] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement with almost perfect retrieval.

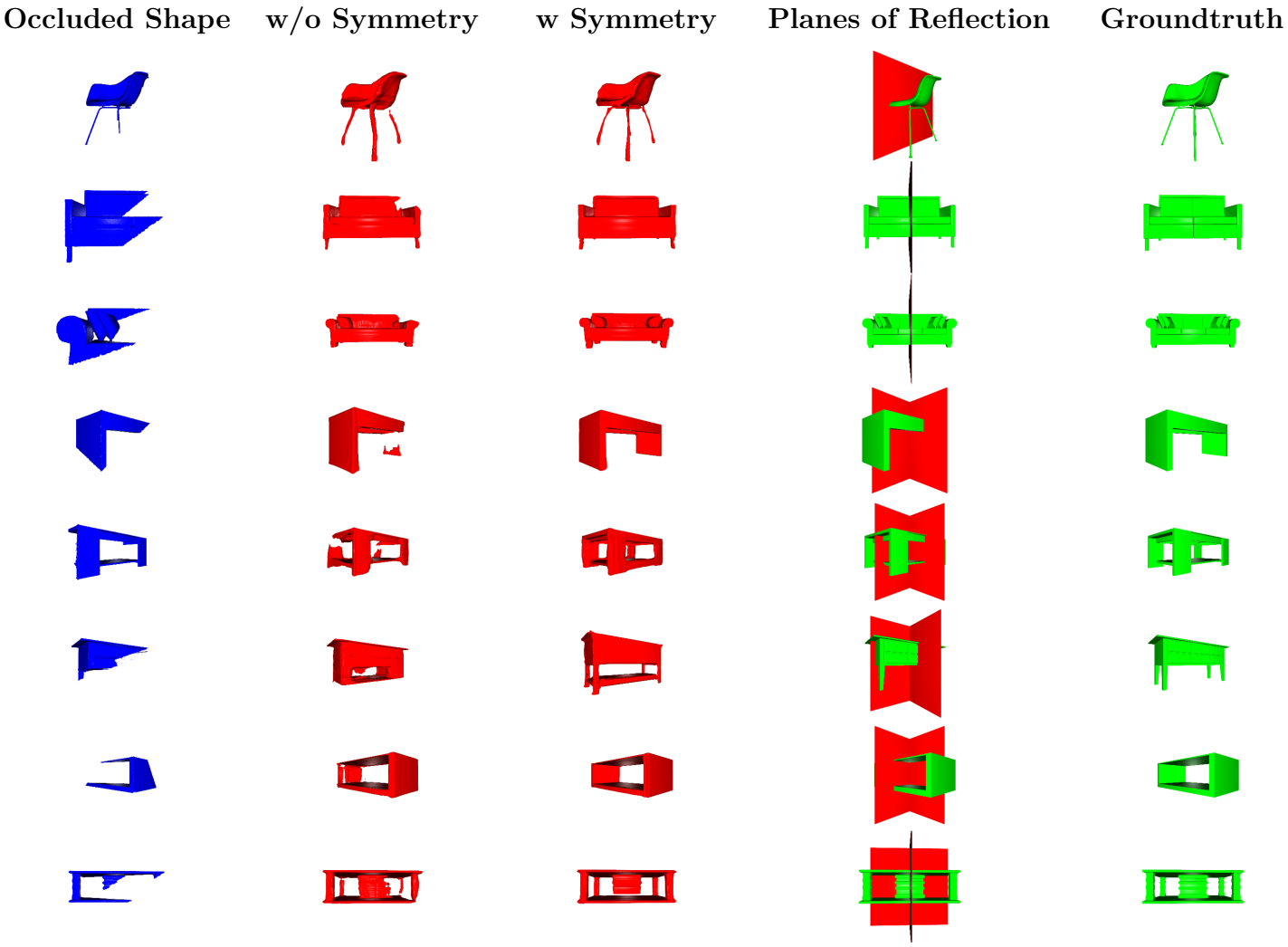

Figure 4.4: Qualitative results showing the benefit of incorporating knowledge of reflective symmetries. Left to right: Partially Occluded input mesh, JSRTE without symmetry information, JSRTE with symmetry information, planes of reflection, groundtruth unoccluded shape. The results incorporating symmetry information provide more accurate and aesthetically pleasing shapes.

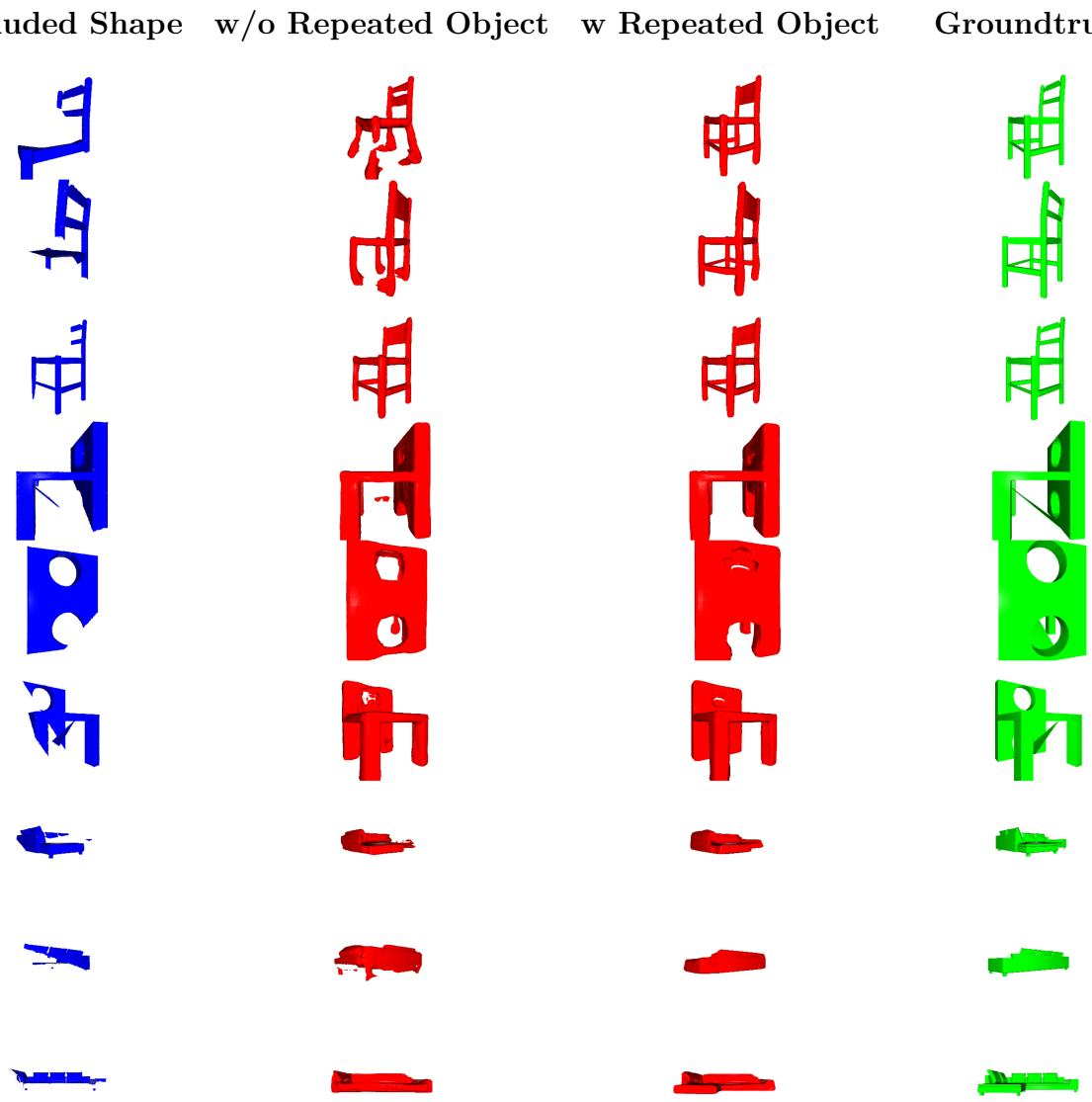

Figure 4.5: Qualitative results indicating the benefit of making use of knowledge about repeated objects. Left to Right: Partially occluded shape, JSRTE solution without using information about repeated objects. JSRTE solution using information about repeated objects, groundtruth. Rows 1-3, 4-6, 7-9, each correspond to multiple instances of the same shape. We observe more robustness by making use of information about repeated objects.

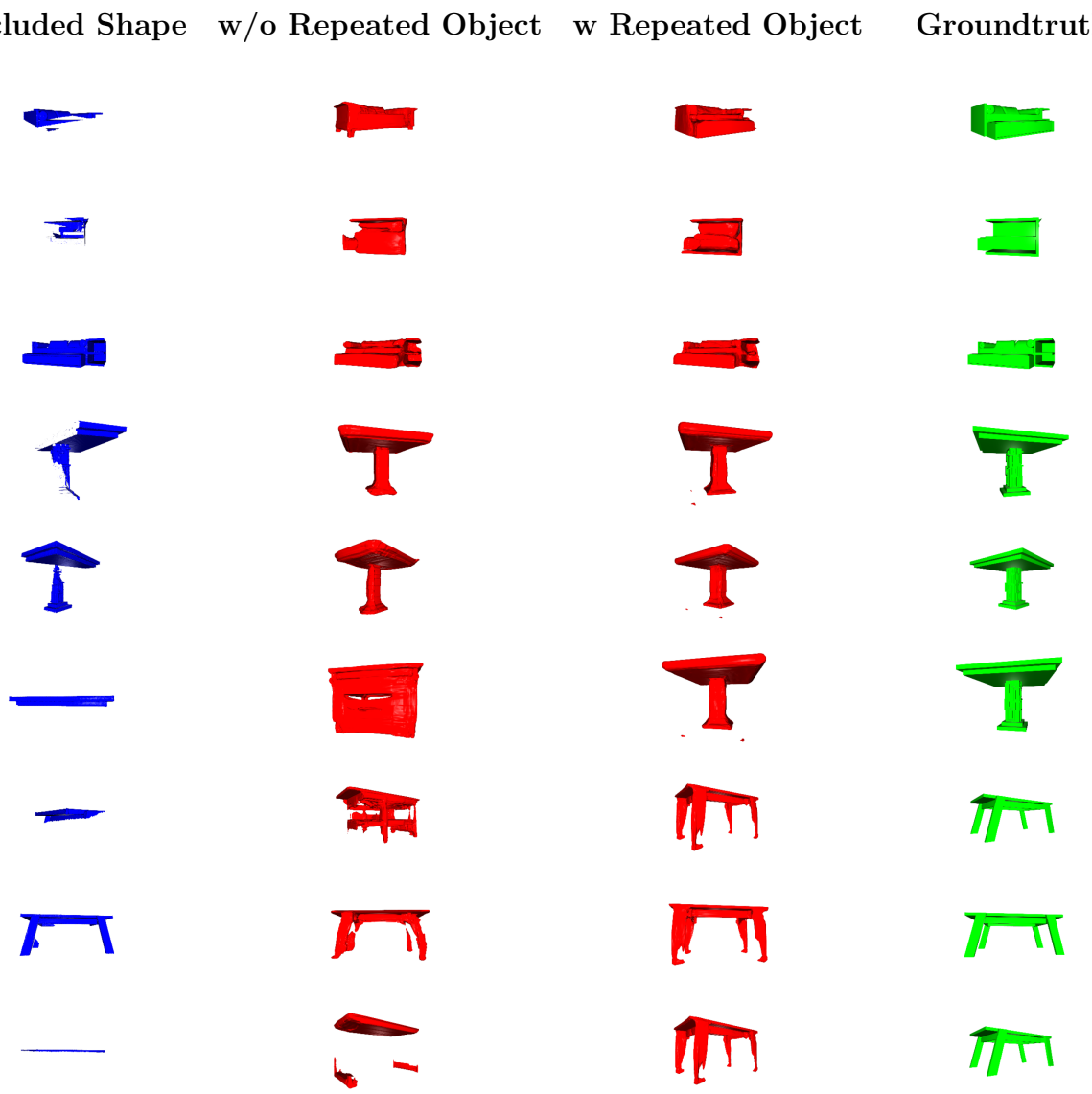

Figure 4.6: Qualitative results indicating the benefit of making use of knowledge about repeated objects. Left to Right: Partially occluded shape, JSRTE solution without using information about repeated objects. JSRTE solution using information about repeated objects, groundtruth. Rows 1-3, 4-6, 7-9, each correspond to multiple instances of the same shape. We observe more robustness by making use of information about repeated objects.

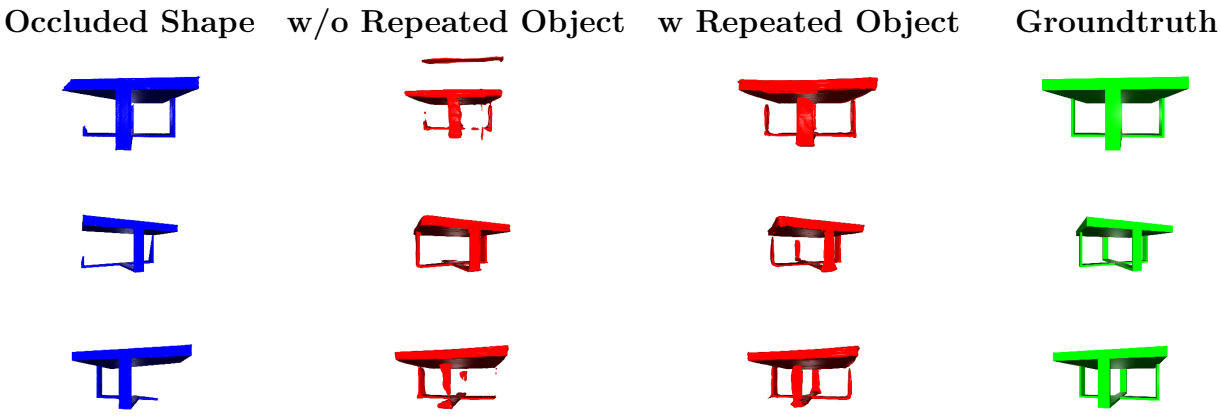

Figure 4.7: Qualitative results indicating the benefit of making use of knowledge about repeated objects. Left to Right: Partially occluded shape, JSRTE solution without using information about repeated objects. JSRTE solution using information about repeated objects, groundtruth. Each row corresponds to a repeated instance of the same object.

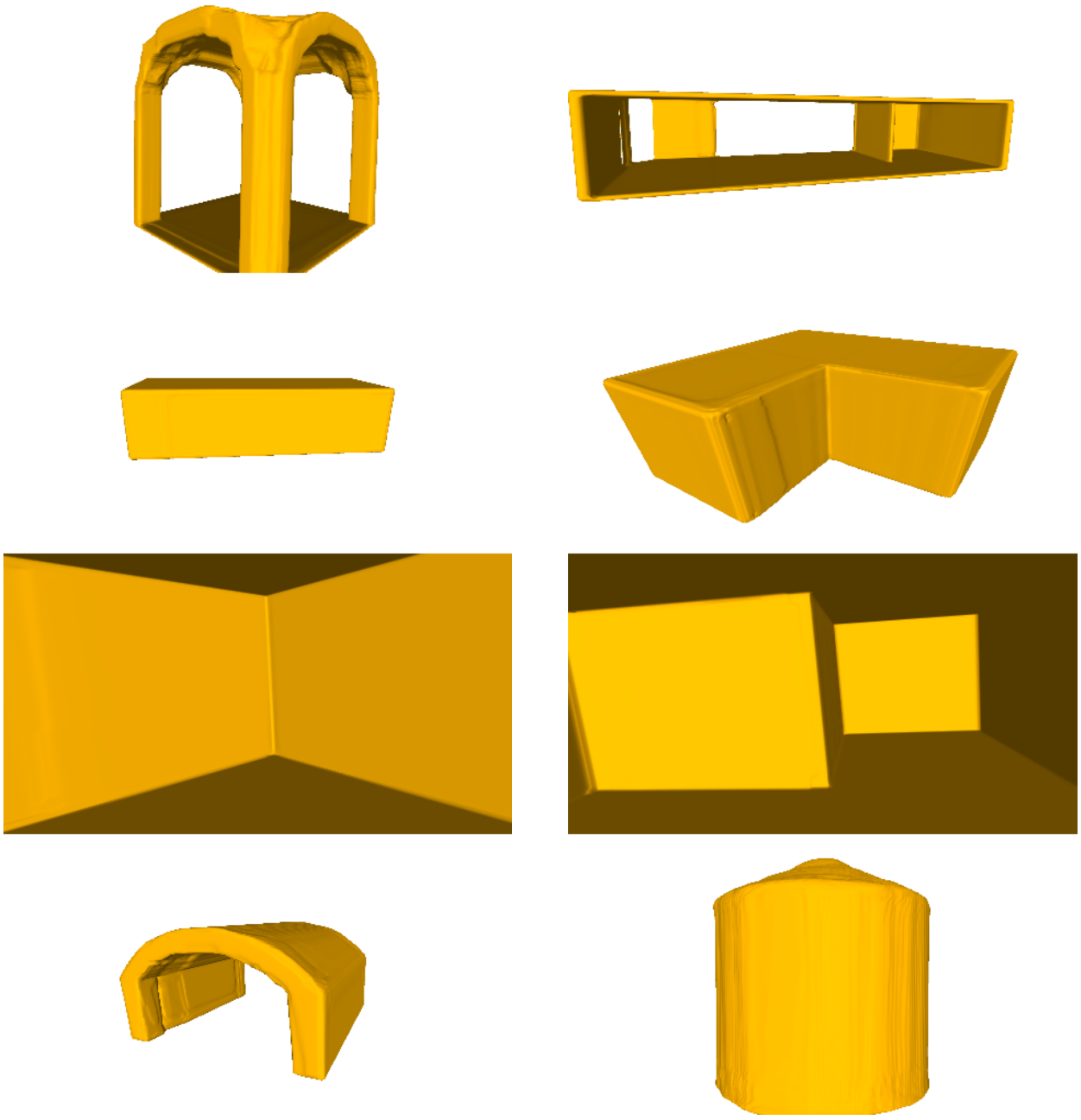

Figure 4.8: Sample images of layouts represented using DeepSDF[76] as a generative model. We are able to represent both planar and non-planar layouts accurately using the same number of parameters.

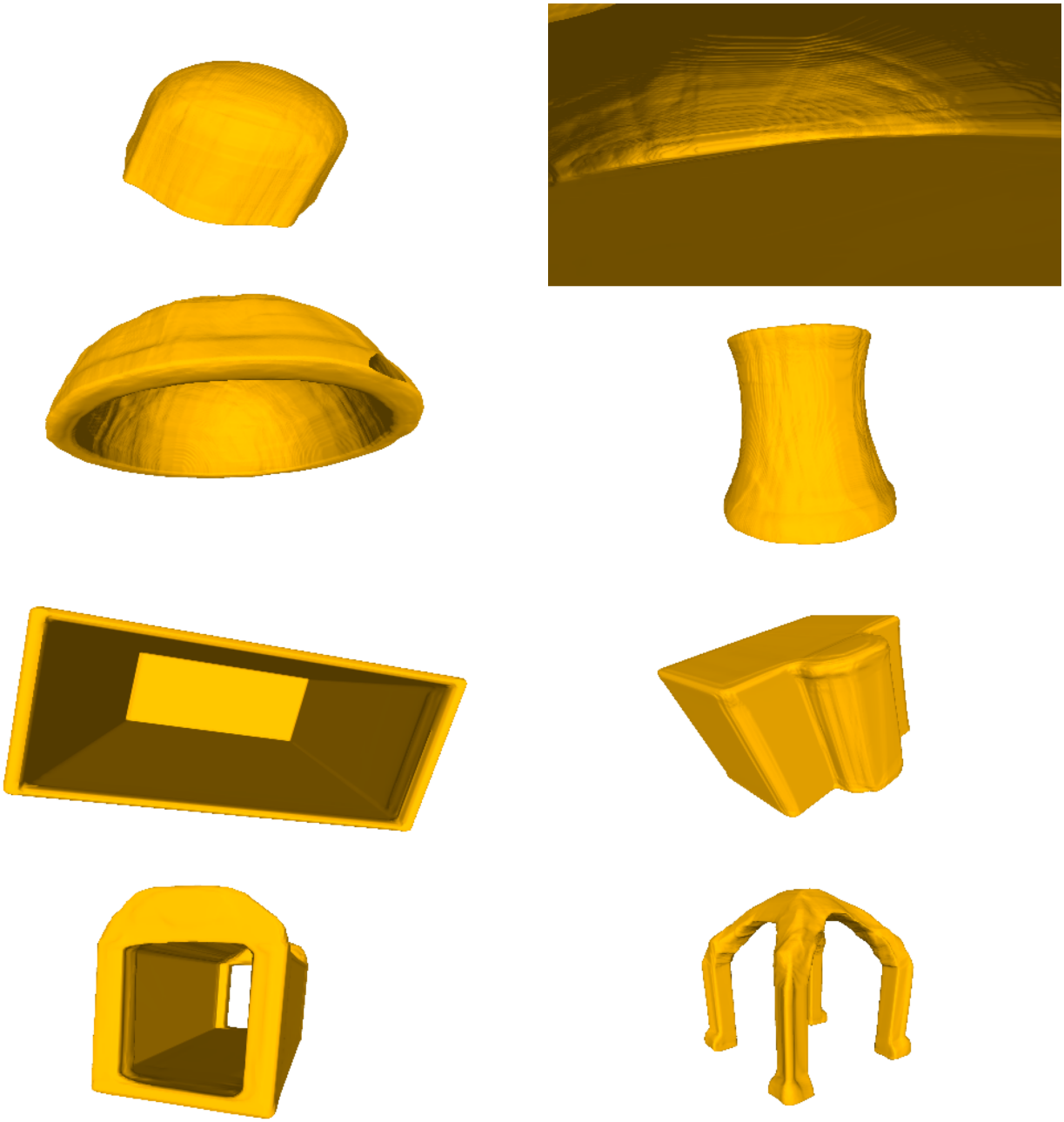

Figure 4.9: Sample images of layouts represented using DeepSDF[76] as a generative model. We are able to represent both planar and non-planar layouts accurately using the same number of parameters.

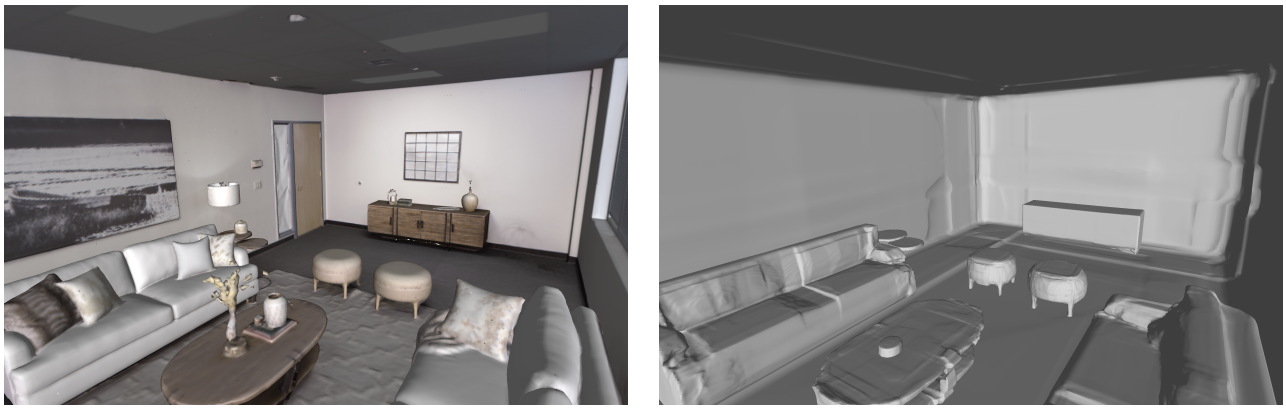

Figure 4.10: Sample results from using the methods proposed in the chapter to model the layout and objects in an indoor scene. Left: 3D scan of an input scene, Right: model generated by our proposed methods.

Figure 4.11: Sample results from using the methods proposed in the chapter to model the layout and objects in an indoor scene. Left: 3D scan of an input scene, Right: model generated by our proposed methods.

Chapter 5

Conclusion and Future Directions

So far in this dissertation, we have presented three data-driven perception algorithms for three different artificially intelligent systems. In chapter 2, we introduced “people as sensors" a data driven perception algorithm for pedestrian detection in autonomous driving scenarios. In chapter 3, we presented an energy disaggregation algorithm that makes use of data-driven models to predict individual appliance energy consumption from whole building energy consumption measurements. Finally, in chapter 4 we presented an algorithm to estimate object shape and pose from data. We showed that this algorithm is also able to provide compact representations for scenes by making use of data-driven shape models.

While the algorithms presented displayed excellent results and showcased the benefit of using data-driven estimation algorithms for perception, they can still be improved to take advantage of upcoming data-driven methods. Some of the datasets and experiments presented were exploratory and designed to illustrate the feasiblity of the algorithms presented. To truly reap the rewards of these algorithms it will be beneficial to design new datasets in some cases and in other cases incorporate these algorithms in larger systems than the ones presented. We now briefly enumerate some interesting future directions and applications of the algorithms presented.

Incorporation of new datasets for Pedestrian Detection in Autonomous Driving

The work presented chapter 2, was tested on the JAAD [83, 84] dataset. While the JAAD dataset provides real world data with pedestrian annotation from frame-to-frame, it was not designed with our use case in mind. For example, it does not provide 3D position data for the pedestrian or camera parameters that would be helpful to extract this data. This prevents us from comparing against and incorporating other predictive models for pedestrian detection. It would be useful to incorporate such datasets into this work to truly understand its benefit for autonomous vehicles.

In addition, it would be beneficial to design control strategies to take advantage of this work for driver safety and comfort in the case of sensor occlusion.

Incorporation of Deep Neural Network based Device Models for Energy Disaggregation

The work presented in chapter 3 may be viewed as a starting point for more sophisticated optimization based approaches for energy disaggregation. The main advantage of the approach presented is its flexibility in allowing the user incorporate proirs into the disaggregation process. However, the iteration presented in chapter 3 makes use of pre-recorded signatures for device modeling. While simple and effective, pre-recorded signatures may be difficult to obtain since it requires human intervention. An alternative approach discussed in the Appendix is to make use of deep neural networks for modeling devices. This alternative has the potential to allow for more expressive device models. In addition, the use of GPU batch processes can make the optimization procedure significantly faster.

Furthermore, the algorithm presented in chapter 3 attempts to estimate the switching times using a tree-like structure. A faster approach would be to use deep neural networks for switching time prediction. A preliminary version of this approach is presented in the Appendix. The benefit of decoupling switching time estimation and device usage prediction is that switching time estimation can be done relatively quickly. Given the switching times, device usage prediction is exponentially faster.

Finally, we note that it would be interesting to characterize the behavior and feasibility of the presented algorithm in scenarios with significantly more devices, since it will then be harder to discriminate between devices. Developing algorithms to deal with such cases is left for future work.

Application of Joint Shape Retrieval and Transform Estimation for Manipulation, Layout Estimation and 3D Reconstruction

The Joint Shape Retrieval and Transform Estimation work presented in chapter 4 proposed a method for estimating object pose and shape from data. One application of this method is for use in robotic manipulation tasks. When the latent space of shapes is also annotated with shape part labels (e.g. cup handle, chair leg) as well as ways to manipulate the objects in the latent space, our method provides a way to generalize this annotation to new objects in new poses and scales. Given this generalization, the task of object manipulation becomes relatively easy.

In addition, we have presented a proof of concept as to how our method can be used for modeling layouts and 3D indoor scenes. Our view of modeling layouts using deep generative models is not the only compact representation possible. In fact for piece-wise planar scenes, more compact representations can be achieved using wireframes [46, 120] or bounded plane segments. It would be advantageous to build a system to explore easy conversion between the various representations.

Moreover, the current dataset used for learning a representation for layouts can be greatly improved. Towards this end, we have started working on a project to capture scans of U.C. Berkeley buildings and annotate them with layout models amongst other useful attributes.

Finally, to ensure that our algorithm works well in broader real world scenes, it is essential to develop robust localization and mapping algorithms that produce relatively clean data to work on. Care must also be taken to reason about clutter in the scene and mis-classified objects.

Bibliography

[1] URL: https://3dwarehouse.sketchup.com/.

[2] Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the 21st ACM International Conference on Machine Learning (ICML)*. 2004, p. 1.

[3] Panos Achlioptas et al. "Learning representations and generative models for 3d point clouds". In: *arXiv preprint arXiv:1707.02392* (2017).

[4] U.S Energy Information Administration. *How much energy is consumed in U.S. residential and commercial buildings?* 2017. URL: https://www.eia.gov/tools/faqs/faq.php?id=86%5C&t=1.

[5] Aitor Aldoma et al. "OUR-CVFH–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6DOF pose estimation". In: *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*. Springer. 2012, pp. 113–122.

[6] Kyle Anderson et al. "BLUED: A Fully Labeled Public Dataset for Event-Based Non-Intrusive Load Monitoring Research". In: *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*. Beijing, China, Aug. 2012.

[7] K. Carrie Armel et al. "Is disaggregation the holy grail of energy efficiency? The case of electricity". In: *Energy Policy* 52 (2013), pp. 213–234. DOI: http://dx.doi.org/10.1016/j.enpol.2012.08.062. URL: http://www.sciencedirect.com/science/article/pii/S0301421512007446.

[8] K Somani Arun, Thomas S Huang, and Steven D Blostein. "Least-squares fitting of two 3-D point sets". In: *IEEE Transactions on pattern analysis and machine intelligence* 5 (1987), pp. 698–700.

[9] Armen Avetisyan, Angela Dai, and Matthias Nießner. "End-to-End CAD Model Retrieval and 9DoF Alignment in 3D Scans". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2551–2560.

[10] Armen Avetisyan et al. "Scan2cad: Learning cad model alignment in rgb-d scans". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2614–2623.

[11] Tirthankar Bandyopadhyay et al. "Intention-aware pedestrian avoidance". In: *Experimental Robotics*. Springer. 2013, pp. 963–977.

[12] Mario E Berges et al. "Enhancing electricity audits in residential buildings with nonintrusive load monitoring". In: *Journal of Industrial Ecology* 14 (2010), pp. 844–858.

[13] Mario Berges et al. "Learning systems for electric consumption of buildings". In: *ASCI International Workshop on Computing in Civil Engineering*. 2009.

[14] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes". In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. International Society for Optics and Photonics. 1992, pp. 586–606.

[15] Andreas Birk and Stefano Carpin. "Merging occupancy grid maps from multiple robots". In: *Proceedings of the IEEE* 94.7 (2006), pp. 1384–1397.

[16] Erik Bylow et al. "Real-time camera tracking and 3D reconstruction using signed distance functions." In: *Robotics: Science and Systems*. Vol. 2. 2013, p. 2.

[17] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[18] Baiming Chen, Ding Zhao, and Huei Peng. "Evaluation of Automated Vehicles Encountering Pedestrians at Unsignalized Crossings". In: *Available on arXiv:1702.00785* (2017).

[19] Sungjoon Choi et al. "A large dataset of object scans". In: *arXiv preprint arXiv:1602.02481* (2016).

[20] Jon Creyts et al. *Reducing U.S. Greenhouse Gas Emissions: How Much at What Cost?* Tech. rep. U.S. Greenhouse Gas Abatement Mapping Initiative, 2007.

[21] Zheng Cui, Sasan Mahmoodi, and Michael Bennett. "A robust and high-performance shape registration technique using characteristic functions". In: *2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS)*. IEEE. 2018, pp. 1–6.

[22] Frank Dellaert, Michael Kaess, et al. "Factor graphs for robot perception". In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 82–95.

[23] Haowen Deng, Tolga Birdal, and Slobodan Ilic. "3D local features for direct pairwise registration". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3244–3253.

[24] Piotr Dollar et al. "Pedestrian detection: An evaluation of the state of the art". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4 (2012), pp. 743–761.

[25] Roy Dong. "New Data Markets Deriving from the Internet of Things: A Societal Perspective on the Design of New Service Models". PhD thesis. UC Berkeley, 2017.

[26] Roy Dong et al. "A dynamical systems approach to energy disaggregation". In: *2013 IEEE 52nd Annu. Conf. on Decision and Control (CDC)*. Dec. 2013, pp. 6335–6340. DOI: 10.1109/CDC.2013.6760891.

[27] Roy Dong et al. "Energy disaggregation via adaptive filtering". In: *2013 51st Annu. Allerton Conf. on Communication, Control, and Computing (Allerton)*. Oct. 2013, pp. 173–180. DOI: 10.1109/Allerton.2013.6736521.

[28] Anup Doshi and Mohan M Trivedi. "Tactical driver behavior prediction and intent inference: A review". In: *14th IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 2011, pp. 1892–1897.

[29] Steven Drenker and Ab Kader. "Nonintrusive monitoring of electric loads". In: *IEEE Computer Applications in Power* 12 (1999), pp. 47–51.

[30] Katherine Driggs Campbell and Ruzena Bajcsy. "Experimental Design for Human-in-the-Loop Driving Simulations". MA thesis. EECS Department, University of California, Berkeley, May 2015.

[31] K. Driggs-Campbell and R. Bajcsy. "Identifying Modes of Intent from Driver Behaviors in Dynamic Environments". In: *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*. July 2015, pp. 739–744.

[32] K. Driggs-Campbell, V. Govindarajan, and R. Bajcsy. "Integrating Intuitive Driver Models in Autonomous Planning for Interactive Maneuvers". In: *IEEE Transactions on Intelligent Transportation Systems* 99 (2017). ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2715836.

[33] Yueqi Duan et al. "Curriculum DeepSDF". In: *arXiv preprint arXiv:2003.08593* (2020).

[34] A. Elfes. "Using occupancy grids for mobile robot perception and navigation". In: *IEEE Computer* 22.6 (June 1989), pp. 46–57. ISSN: 0018-9162. DOI: 10.1109/2.30720.

[35] Ehsan Elhamifar and Shankar Sastry. "Energy Disaggregation via Learning Powerlets and Sparse Coding." In: *AAAI*. 2015, pp. 629–635.

[36] Linda Farinaccio and Radu Zmeureanu. "Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses". In: *Energy and Buildings* 30 (1999), pp. 245–259.

[37] Michael Garland and Paul S Heckbert. "Surface simplification using quadric error metrics". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997, pp. 209–216.

[38] Sukru Yaren Gelbal et al. "Elastic band based pedestrian collision avoidance using V2X communication". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2017.

[39] Kyle Genova et al. "Learning shape templates with structured implicit functions". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7154–7164.

[40] Nicolas Guéguen, Sébastien Meineri, and Chloé Eyssartier. "A pedestrian's stare and drivers' stopping behavior: A field experiment at the pedestrian crossing". In: *Safety Science* 75 (2015), pp. 87–89.

[41] J. Guivant and E. Nebot. "Improving computational and memory requirements of simultaneous localization and map building algorithms". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3. 2002, pp. 2731–2736. DOI: 10.1109/ROBOT.2002.1013645.

[42] Yulan Guo et al. "3D object recognition in cluttered scenes with local surface features: a survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), pp. 2270–2287.

[43] G.W. Hart. "Nonintrusive appliance load monitoring". In: *Proc. of the IEEE* 80.12 (1992), pp. 1870–1891. ISSN: 0018-9219. DOI: 10.1109/5.192069.

[44] Berthold KP Horn. "Closed-form solution of absolute orientation using unit quaternions". In: *Josa a* 4.4 (1987), pp. 629–642.

[45] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. "Closed-form solution of absolute orientation using orthonormal matrices". In: *JOSA A* 5.7 (1988), pp. 1127–1135.

[46] Kun Huang et al. "Learning to parse wireframes in images of man-made environments". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 626–635.

[47] Xiaolei Huang, Nikos Paragios, and Dimitris N Metaxas. "Shape registration in implicit spaces using information theory and free form deformations". In: *IEEE transactions on pattern analysis and machine intelligence* 28.8 (2006), pp. 1303–1318.

[48] Hamid Izadinia, Qi Shan, and Steven M Seitz. "Im2cad". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5134–5143.

[49] Krishna Murthy J. et al. "Kaolin: A PyTorch Library for Accelerating 3D Deep Learning Research". In: *arXiv:1911.05063* (2019).

[50] Matthew J. Johnson and Alan S. Willsky. "Bayesian Nonparametric Hidden Semi-Markov Models". In: *arXiv:1203.1365* (Mar. 2012). eprint: 1203.1365.

[51] Immanuel Kant. *The critique of pure reason, Project Gutenberg, 2007*.

[52] Sagi Katz, Ayellet Tal, and Ronen Basri. "Direct visibility of point sets". In: *ACM SIGGRAPH 2007 papers*. 2007, 24–es.

[53] Michael Kazhdan et al. "A reflective symmetry descriptor". In: *European Conference on Computer Vision*. Springer. 2002, pp. 642–656.

[54] Jack Kelly and William Knottenbelt. "Neural NILM: Deep neural networks applied to energy disaggregation". In: *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. ACM. 2015, pp. 55–64.

[55] Jack Kelly and William Knottenbelt. "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes". In: *Scientific Data* 2.150007 (). DOI: 10.1038/sdata.2015.7.

[56] Jack Kelly et al. "NILMTK v0.2: A Non-intrusive Load Monitoring Toolkit for Large Scale Data Sets". In: *The first ACM Workshop On Embedded Systems For Energy-Efficient Buildings at BuildSys 2014*. Memphis, USA, 2014.

[57] H. Kim et al. "Unsupervised Disaggregation of Low Frequency Power Measurements". In: *SDM'11*. 2011, pp. 747–758.

[58] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[59] J. Zico Kolter and Tommi Jaakkola. "Approximate Inference in Additive Factorial HMMs with Application to Energy Disaggregation". In: *Proc. of the Int. Conf. on Artificial Intelligence and Statistics*. 2012.

[60] J. Zico Kolter and Matthew J. Johnson. "REDD: A Public Data Set for Energy Disaggregation Research". In: *Proc. of the SustKDD Workshop on Data Mining Applications in Sustainability*. 2011.

[61] J. Zico Kolter and Andrew Y. Ng. "Energy Disaggregation via Discriminative Sparse Coding". In: *Neural Information Processing Systems*. 2010.

[62] Iuliia Kotseruba, Amir Rasouli, and John K Tsotsos. "Joint attention in autonomous driving (JAAD)". In: *Available on arXiv:1609.04741* (2016).

[63] John A. Laitner, Karen Ehrhardt-Martinez, and Vanessa McKinney. "Examining the scale of the Behaviour Energy Efficiency Continuum". In: *European Council for an Energy Efficient Economy*. 2009.

[64] Henning Lange and Mario Bergés. "The neural energy decoder: energy disaggregation by combining binary subcomponents". In: *NILM2016 3rd international workshop on non-intrusive load monitoring. Retrieved from nilmworkshop. org Google Scholar*. 2016.

[65] Liangda Li and Hongyuan Zha. "Energy Usage Behavior Modeling in Energy Disaggregation via Hawkes Processes". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 9.3 (2018), p. 36.

[66] Yangyan Li et al. "Database-assisted object retrieval for real-time 3d reconstruction". In: *Computer Graphics Forum*. Vol. 34. 2. Wiley Online Library. 2015, pp. 435–446.

[67] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM siggraph computer graphics* 21.4 (1987), pp. 163–169.

[68] Yi Ma et al. *An invitation to 3-d vision: from images to geometric models*. Vol. 26. Springer Science & Business Media, 2012.

[69] Sasan Mahmoodi, Muayed S Al-Huseiny, and Mark S Nixon. "Similarity registration for shapes based on signed distance functions". In: *International Symposium on Visual Computing*. Springer. 2012, pp. 599–609.

[70] Stephen Makonin et al. "Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014". In: *Scientific data* 3 (2016).

[71] Stephen Makonin et al. "Exploiting hmm sparsity to perform online real-time nonintrusive load monitoring". In: *IEEE Transactions on Smart Grid* 7.6 (2016), pp. 2575–2585.

[72] Lukas Mauch and Bin Yang. "A novel DNN-HMM-based approach for extracting single loads from aggregate power signals". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 2384–2388.

[73] Hiroshi Mitsumoto et al. "3-D reconstruction using mirror images based on a plane symmetry recovering method". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 9 (1992), pp. 941–946.

[74] Robert Paolini et al. "A data-driven statistical framework for post-grasp manipulation". In: *The International Journal of Robotics Research* 33.4 (2014), pp. 600–615.

[75] Nikos Paragios, Mikael Rousson, and Visvanathan Ramesh. "Non-rigid registration using distance functions". In: *Computer Vision and Image Understanding* 89.2-3 (2003), pp. 142–165.

[76] Jeong Joon Park et al. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[77] Oliver Parson et al. "Nonintrusive load monitoring using prior models of general appliance types". In: *Proc. of the 26th AAAI Conf. on Artificial Intelligence*. 2012.

[78] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[79] Sundeep Pattem. "Unsupervised Disaggregation for Non-intrusive Load Monitoring". In: *11th International Conference on Machine Learning and Applications (ICMLA), 2012*. Vol. 2. IEEE. 2012, pp. 515–520.

[80] Quang-Hieu Pham et al. "SHREC'18: Rgb-d object-to-cad retrieval". In: *Proc. 3DOR* 2 (2018).

[81] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[82] Dwi Rahayu et al. “Learning to be energy-wise: discriminative methods for load disaggregation”. In: *Third International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012*. IEEE. 2012, pp. 1–4.

[83] A. Rasouli, I. Kotseruba, and J. K. Tsotsos. “Agreeing to cross: How drivers and pedestrians communicate”. In: *IEEE Intelligent Vehicles Symposium (IV)*. June 2017, pp. 264–269. DOI: 10.1109/IVS.2017.7995730.

[84] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. “Are they going to cross? A benchmark dataset and baseline for pedestrian crosswalk behavior”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 206–213.

[85] Olinde Rodrigues. “De l’attraction des sphéroı¨des, Correspondence sur l’É-cole Impériale Polytechnique”. PhD thesis. PhD thesis, Thesis for the Faculty of Science of the University of Paris, 1816.

[86] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. “Fast point feature histograms (FPFH) for 3D registration”. In: *2009 IEEE international conference on robotics and automation*. IEEE. 2009, pp. 3212–3217.

[87] Radu Bogdan Rusu and Steve Cousins. “3d is here: Point cloud library (pcl)”. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 1–4.

[88] Katayoun Salamati et al. “Event-based modeling of driver yielding behavior to pedestrians at two-lane roundabout approaches”. In: *Transportation Research Record: Journal of the Transportation Research Board* 2389 (2013), pp. 1–11.

[89] Samuele Salti, Federico Tombari, and Luigi Di Stefano. “SHOT: Unique signatures of histograms for surface and texture description”. In: *Computer Vision and Image Understanding* 125 (2014), pp. 251–264.

[90] Ying Shan et al. “Linear model hashing and batch ransac for rapid and accurate object recognition”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 2. IEEE. 2004, pp. II–II.

[91] Huijuan Shao, Manish Marwah, and Naren Ramakrishnan. “A Temporal Motif Mining Approach to Unsupervised Energy Disaggregation”. In: *Proceedings of 1st International Non-Intrusive Load Monitoring Workshop*. 2012.

[92] V. A. Shia et al. “Semiautonomous Vehicular Control Using Driver Modeling”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.6 (Dec. 2014), pp. 2696–2709. ISSN: 1524-9050. DOI: 10.1109/TITS.2014.2325776.

[93] Miroslava Slavcheva. “Signed Distance Fields for Rigid and Deformable 3D Reconstruction”. PhD thesis. Technische Universität München, 2018.

[94] Miroslava Slavcheva et al. “Sdf-2-sdf: Highly accurate 3d object reconstruction”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 680–696.

[95] Shuran Song et al. "Semantic Scene Completion from a Single Depth Image". In: *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition* (2017).

[96] Pablo Speciale et al. "A symmetry prior for convex variational 3d reconstruction". In: *European Conference on Computer Vision*. Springer. 2016, pp. 313–328.

[97] Marc Steiner. "Statistical Shape Models with Signed Distance Functions". In: ().

[98] Julian Straub et al. "The Replica Dataset: A Digital Replica of Indoor Spaces". In: *arXiv preprint arXiv:1906.05797* (2019).

[99] Rouxian Sun et al. "The estimation of vehicle speed and stopping distance by pedestrians crossing streets in a naturalistic traffic environment". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 30 (2015), pp. 97–106.

[100] Kosuke Suzuki et al. "Nonintrusive appliance load monitoring based on integer programming". In: *SICE Annual Conference, 2008*. IEEE. 2008, pp. 2742–2747.

[101] Maxim Tatarchenko et al. "What Do Single-view 3D Reconstruction Networks Learn?" In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3405–3414.

[102] *The Boundary of a Set under Homeomorphisms on Topological Spaces*. URL: `http://mathonline.wikidot.com/the-boundary-of-a-set-under-homeomorphisms-on-topological-sp`.

[103] *The Interior of a Set under Homeomorphisms on Topological Spaces*. URL: `http://mathonline.wikidot.com/the-interior-of-a-set-under-homeomorphisms-on-topological-sp`.

[104] Sebastian Thrun. "Learning occupancy grids with forward models". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*. Vol. 3. IEEE. 2001, pp. 1676–1681.

[105] Sebastian Thrun et al. "Robotic mapping: A survey". In: *Exploring Artificial Intelligence in the New Millennium* 1 (2002), pp. 1–35.

[106] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT press, 2005, pp. 285–289.

[107] Federico Tombari and Luigi Di Stefano. "Object recognition in 3d scenes with occlusions and clutter by hough voting". In: *2010 Fourth Pacific-Rim Symposium on Image and Video Technology*. IEEE. 2010, pp. 349–355.

[108] Federico Tombari, Samuele Salti, and Luigi Di Stefano. "Unique signatures of histograms for local surface description". In: *European conference on computer vision*. Springer. 2010, pp. 356–369.

[109] Kenneth E Train. *Discrete choice methods with simulation*. Cambridge University Press, 2009.

[110] Shubham Tulsiani et al. "Factoring shape, pose, and layout from the 2d image of a 3d scene". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 302–310.

[111] Shinji Umeyama. "Least-squares estimation of transformation parameters between two point patterns". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 4 (1991), pp. 376–380.

[112] Michael P Vandenbergh, Jack Barkenbus, and Jonathan Gilligan. "Individual carbon emissions: The low-hanging fruit". In: *UCLA L. Rev.* 55 (2007), p. 1701.

[113] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: https://doi.org/10.1038/s41592-019-0686-2.

[114] Michael W Walker, Lejun Shao, and Richard A Volz. "Estimating 3-D location parameters using dual number quaternions". In: *CVGIP: image understanding* 54.3 (1991), pp. 358–367.

[115] Markus Weiss et al. "Leveraging smart meter data to recognize home appliances". In: *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*. IEEE. 2012, pp. 190–197.

[116] Allen Y Yang et al. "Symmetry-based 3-D reconstruction from perspective images". In: *Computer Vision and Image Understanding* 99.2 (2005), pp. 210–240.

[117] Andy Zeng et al. "3dmatch: Learning local geometric descriptors from rgb-d reconstructions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1802–1811.

[118] Chaoyun Zhang et al. "Sequence-to-point learning with neural networks for non-intrusive load monitoring". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[119] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[120] Yichao Zhou, Haozhi Qi, and Yi Ma. "End-to-end wireframe parsing". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 962–971.

[121] Ahmed Zoha et al. "Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey". In: *Sensors* 12.12 (2012), pp. 16838–16866.

Appendix A

Supplementary Results for Chapter 3

A.1 Proof of Proposition 3.3.1

By definition,

$$\widehat{u_{\mathrm{MAP}}} = \max_u P_{\mathcal{U}}(u) P_{\mathcal{Z}|\mathcal{U}}(z|u) \ . \tag{A.1}$$

First, we seek to show that:

$$T_{switch}(\widehat{u_{\mathrm{MAP}}}) = \arg\max_{T_{switch}} s(T_{switch}|z) \ .$$

Notice that:

$$\max_{T_{switch}} \ s(T_{switch}|z) = \max_u P_{\mathcal{U}}(u) P_{\mathcal{Z}|\mathcal{U}}(z|u) \ ,$$

since the value of T_{switch} is unrestricted in the above optimization.
Also observe that:

$$\max_{T_{switch}=T_{switch}(\widehat{u_{\mathrm{MAP}}})} s(T_{switch}|z) = \max_{T_{switch}=T_{switch}(\widehat{u_{\mathrm{MAP}}})} \ \max_{u: \ T_{switch}(u)=T_{switch}} P_{\mathcal{U}}(u) P_{\mathcal{Z}|\mathcal{U}}(z|u) \ ,$$

and since by definition $\widehat{u_{\mathrm{MAP}}}$ is a global optimizer,

$$\begin{aligned} &\max_{T_{switch}=T_{switch}(\widehat{u_{\mathrm{MAP}}})} \ \max_{u: \ T_{switch}(u)=T_{switch}} P_{\mathcal{U}}(u) P_{\mathcal{Z}|\mathcal{U}}(z|u) \ = \max_u P_{\mathcal{U}}(u) P_{\mathcal{Z}|\mathcal{U}}(z|u) \\ &= \max_{T_{switch}} \ s(T_{switch}|z) \end{aligned} \ .$$

Therefore,

$$\max_{T_{switch}=T_{switch}(\widehat{u_{\mathrm{MAP}}})} s(T_{switch}|z) = \max_{T_{switch}} \ s(T_{switch}|z)$$

$$\implies T_{switch}(\widehat{u_{\text{MAP}}}) = \arg\max_{T_{switch}} s(T_{switch}|z) \ . \quad \square$$

Next, we want to show that if $T_{switch}(\widehat{u_{\text{MAP}}})$ is known, then:

$$\widehat{u_{\text{MAP}}} = \arg\max_{\substack{u: \\ T_{switch}(u)=T_{switch}(\widehat{u_{\text{MAP}}})}} \Big(P_{\mathcal{U}}(u)P_{\mathcal{Z}|\mathcal{U}}(z|u)\Big) \ .$$

Indeed, if $T_{switch}(\widehat{u_{\text{MAP}}})$ is known, then:

$$\widehat{u_{\text{MAP}}} \in \{u : T_{switch}(u) = T_{switch}(\widehat{u_{\text{MAP}}})\} \ .$$

Since $\widehat{u_{\text{MAP}}}$ is a global optimizer for $P_{\mathcal{U}}(u)P_{\mathcal{Z}|\mathcal{U}}(z|u)$, then it must be that:

$$\widehat{u_{\text{MAP}}} = \arg\max_{\substack{u: \\ T_{switch}(u)=T_{switch}(\widehat{u_{\text{MAP}}})}} \Big(P_{\mathcal{U}}(u)P_{\mathcal{Z}|\mathcal{U}}(z|u)\Big) \ . \quad \square$$

Finally, we show that:

$$P_{\mathcal{U}}(\widehat{u_{\text{MAP}}})P_{\mathcal{Z}|\mathcal{U}}(z|\widehat{u_{\text{MAP}}}) = \max_{T_{switch}} s(T_{switch}|z) \ .$$

We can establish by (3.6) that:

$$s(T_{switch}(\widehat{u_{\text{MAP}}})|z) = \max_{T_{switch}} s(T_{switch}|z) \ . \tag{A.2}$$

We can also establish by Definition (3.3.2) , (3.7) and since $\widehat{u_{\text{MAP}}} \in \{u : T_{switch}(u) = T_{switch}(\widehat{u_{\text{MAP}}})\}$, that:

$$s(T_{switch}(\widehat{u_{\text{MAP}}})|z) = P_{\mathcal{U}}(\widehat{u_{\text{MAP}}})P_{\mathcal{Z}|\mathcal{U}}(z|\widehat{u_{\text{MAP}}}) \ . \tag{A.3}$$

Then by (A.2) and (A.3),

$$P_{\mathcal{U}}(\widehat{u_{\text{MAP}}})P_{\mathcal{Z}|\mathcal{U}}(z|\widehat{u_{\text{MAP}}}) = \max_{T_{switch}} s(T_{switch}|z) \ . \quad \square$$

A.2 Proof of Equation 3.11

$$P_{\mathcal{Z}|\mathcal{U}}(z|u) = \int \cdots \int \left[\left(\prod_{i=1}^{D-1} P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}(y^{(i)}|u^{(i)}) \right) P_{\mathcal{Y}^{(D)}|\mathcal{U}^{(D)}} \left(z - \sum_{i=1}^{D-1} y^{(i)} \middle| u^{(D)} \right) \right] dy^{(1)} \dots dy^{(D-1)} \quad \text{(A.4)}$$

$$= \int \cdots \int \left[\left(\prod_{i=1}^{D-1} \prod_{n=0}^{N} P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}{}_{s_n}}(y^{(i)}[s_n]|u^{(i)}[t_n]) \right) \times \prod_{n=0}^{N} P_{\mathcal{Y}^{(D)}|\mathcal{U}^{(D)}{}_{s_n}} \left(z[s_n] - \sum_{i=1}^{D-1} y^{(i)}[s_n] \middle| u^{(D)}[t_n] \right) \right] dy^{(1)}[s_0] dy^{(1)}[s_1] \dots dy^{(D-1)}[s_N]] \quad \text{(A.5)}$$

$$= \prod_{n=0}^{N} \int \cdots \int \prod_{i=1}^{D-1} P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}{}_{s_n}}(y^{(i)}[s_n]|u^{(i)}[t_n]) \times P_{\mathcal{Y}^{(D)}|\mathcal{U}^{(D)}{}_{s_n}} \left(z[s_n] - \sum_{i=1}^{D-1} y^{(i)}[s_n] \middle| u^{(D)}[t_n] \right) dy^{(1)}[s_n] \dots dy^{(D-1)}[s_n] \quad \text{(A.6)}$$

$$= \prod_{n=0}^{N} (P_{\mathcal{Y}^{(1)}|\mathcal{U}^{(1)}{}_{s_n}} * P_{\mathcal{Y}^{(2)}|\mathcal{U}^{(2)}{}_{s_n}} * \ldots * P_{\mathcal{Y}^{(D)}|\mathcal{U}^{(D)}{}_{s_n}}(z[s_n]|u[t_n])) \quad \text{(A.7)}$$

$$= \prod_{n=0}^{N} P_{\mathcal{Z}|\mathcal{U}_{s_n}}(z[s_n]|u[t_n]) \ . \quad \square \quad \text{(A.8)}$$

A.3 Proof of Proposition 3.4.1

Making use of (3.9) and taking the log of the right side of (3.6), we get that:

$$\widehat{u_{\text{MAP}}} = \arg\max_{u} \sum_{n=0}^{N} \left[\ln \left(P_{\mathcal{U}_{s_n}}(u[t_n]) \right) + \ln \left(P_{\mathcal{Z}|\mathcal{U}_{s_n}}(z[s_n]|u[t_n]) \right) \right] \ .$$

The optimizer remains the same since the log function is a monotonic increasing function.

In addition, since the values of u in each segment do not affect each other, the optimization decouples so that the optimal value of the objective is given as:

$$\widehat{u_{\text{MAP}}} = \sum_{n=0}^{N} \max_{u[t_n]} \left[\ln \left(P_{\mathcal{U}_{s_n}}(u[t_n]) \right) + \ln \left(P_{\mathcal{Z}|\mathcal{U}_{s_n}}(z[s_n]|u[t_n]) \right) \right] \ .$$

Then $\widehat{u_{\text{MAP}}}$ must be such that,

$$\widehat{u_{\text{MAP}}}[t_n] = \arg\max_u \left[\ln\left(P_{\mathcal{U}_{s_n}}(u[t_n])\right) + \ln\left(P_{\mathcal{Z}|\mathcal{U}_{s_n}}(z[s_n]|u[t_n])\right)\right] \ \forall n \ . \quad \square$$

A.4 Complementary Approach to Energy Disaggregation using Deep Neural Networks.

Recently, the energy disaggregation community has witnessed a rise in the use of deep neural networks as a tool for disaggregation. While these approaches provide impressive results, it is not very clear how to incorporate priors into neural network based disaggreagtion methods. In addition, since most approaches rely on regressing a function to predict device energy consumption, it can be unclear if the predictions still possess desirable properties (for example, that the sum of device power consumption is equal to the aggregate consumption or that power values are non-negative) on new data.

The framework presented in Chapter 3, provides a complementary approach that can be used along with deep neural networks to incorporate user priors as well as approximately satisfy constraints leading to desirable properties of the predicted signal. In particular, we may make use of deep neural networks to model devices and then incorporate these models into the framework we provided.

Device modeling using Deep Neural Networks

For the i'th device, given N input-output pairs of historical data $\{(u_j^{(i)}, y_j^{(i)})\}_{j=1}^N$, we seek to find a function $f^{(i)}(u^i, \theta)$ that solves the following optimization problem:

$$\min_\theta \ \sum_{j=1}^{N} |f^{(i)}(u_j^{(i)}, \theta) - y_j^{(i)}|.$$

We model f using a deep neural network parameterized by θ, and assume that future observations of input-output pairs are drawn from the same data generating process as those observed in historical data. We may then set the device model for the i'th device as $\mathcal{H}^{(i)} = f^{(i)}$. Further, we assume that the observed output for each model may be modeled as:

$$y^{(i)} = f(u^{(i)}, \theta) + \mathcal{W}^{(i)}$$

where $\mathcal{W}^{(i)}$ is a noise variable modeled as a random vector with a known distribution. This implicitly defines a distribution for $\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}$ and hence $P_{\mathcal{Y}^{(i)}|\mathcal{U}^{(i)}}(y^{(i)}|u^{(i)})$ is known. The rest of the formulation then follows as described in chapter 3.

Implementation details

In practice, most available datasets do not contain the input signal in addition to the measured output signal per device. To ameliorate this situation, we assume that each device has a fixed number of modes and that the outputs is approximately constant for each mode. We then make use of k-means clustering to estimate these modes per device. We set the input for each mode to correspond to the integer index representing the mode. In this fashion we are able to extract input-output pairs for each device.

We make use of a siz layer neural network described in Table A.1 below and implemented in PyTorch[78]. We train using Adam [58] with a learning rate of 0.01 and a rate decay every 10 epochs. We train for a total of 30 epochs.

Table A.1: Architecture for Convolutional Neural Network used to model devices. Each column describes the kernel width and height, as well as the number of channels and replication padding respectively. Batch normalization, a ReLU layer and a Dropout layer with probability 0.3 are placed between the layers shown below.

	Layer1	Layer2	Layer3	Layer4	Layer5	Layer6	Non-negative Activation
Input	10x1, 30, (4,5)	8x1, 30, (3,4)	6x1, 40, (2,3)	5x1, 50, (2,2)	5x1, 50, (2,2)	1x1, 1, (0,0)	ReLU

The Rectifying Linear Unit (ReLU) at the end of the network is used to prevent negative power predictions. In addition, we train our model on subsequences of input and output pairs so that it can be used in an iterative manner to predict the output at the current time given all input from previous time steps. We pad the subsequences with -1 until them reach the length of the time horizon T before feeding it to the network.

Switching Times prediction using Deep Neural Networks

Recall that in chapter 3, we defined switching times as the following: For some fixed $v : \{0, 1, \ldots, T\} \to \mathbb{R}^D$, we can define the *switching times* of v, denoted $T_{switch}(v) \subset \{0, 1, \ldots, T\}$, as the unique set such that:

- $v[t-1] \neq v[t]$ for all $t \in T_{switch}(v)$.
- $v[t-1] = v[t]$ for all $t \notin T_{switch}(v)$.

We also suggested that given a switching time, solving the energy dissagregation problem becomes significantly easier. Rather than exploring a tree-like structure to estimate the switching times as suggested in chapter 3, an alternative approach is to predict the optimal switching times and then solve the optimization problem. Essentially, this saves computational resources for time steps not in the set of optimal switching times. In addition, this is a reasonable approach since it may be relatively easy to detect when changes occur in the aggregate signal. To extend this further, we may predict not only the switching times for the aggregate signal but also for the individual devices, given the aggregate signal. The switching times for the aggregate signal will then be the union of switching times for the individual devices. The benefit of this is that we may use per device switching times as a prior to help us discriminate between predictions where the power signal for different devices are relatively similar.

To accomplish this task, we make use of a deep neural network to predict per device switching times, given the aggregate energy consumption signal. Essentially we treat it as a sequence-to-sequence prediction task, where we predict a sequence of 0's and 1's given the aggregate signal. Each value in the prediction corresponds to a classification of whether or not we believe that corresponding index is a switching time. Table A.2, below details the architecture of our switching times prediction network. We have made use of a sigmoid activation in the final layer to restrict the outputs to be in $(0, 1)$. We then threshold the values, accepting time steps with value greater than 0.2 as a predicted switching time.

Table A.2: Architecture for Convolutional Neural Network used to predict per device switching times. Each column describes the kernel width and height, as well as the number of channels and zero padding respectively. Batch normalization, a ReLU layer and a Dropout layer with probability 0.3 are placed between each of the layers shown above. In the final layer D denotes the total number of devices as defined in chapter 3.

	Layer1	Layer2	Layer3	Layer4	Layer5	Activation
Input	11x1, 30, (5,5)	9x1, 30, (4,4)	7x1, 30, (3,3)	5x1, 10, (2,2)	5x1, D, (2,2)	Sigmoid

To generate training data for our network, we obtain the groundtruth device energy consumption and make use of per device thresholding for change point detection. These change points are then labelled as the groundtruth switching times.

Experiments

We train both networks on subsets of the AMPds dataset [70] and perform experiments to test the viability of our suggestions. We choose a subset of devices from the AMPds dataset[70], namely the Clothes Dryer (CDE), Dishwasher (DWE), Fridge (FGE) and Heat Pump (HPE) and create an artificial aggregate signals by summing up the individual power consumptions. We train our device models and switching times predictor on a subset of this data and perform disaggregation on a disjoint subset of the data using the formulation presented in chapter 3. The only difference being that we assume we already have an optimal switching time (predicted by the switching times prediction network). For time steps not included in the switching times, we retain the input value at the last switching time, this significantly improves the computational speed. Fig. A.1 and Fig, A.2 below, show some sample results validating our approach. While these results and the ability to include arbitrary prior in the disaggregation problem are promising, there are still questions with regards to which priors are important and when they should be used.

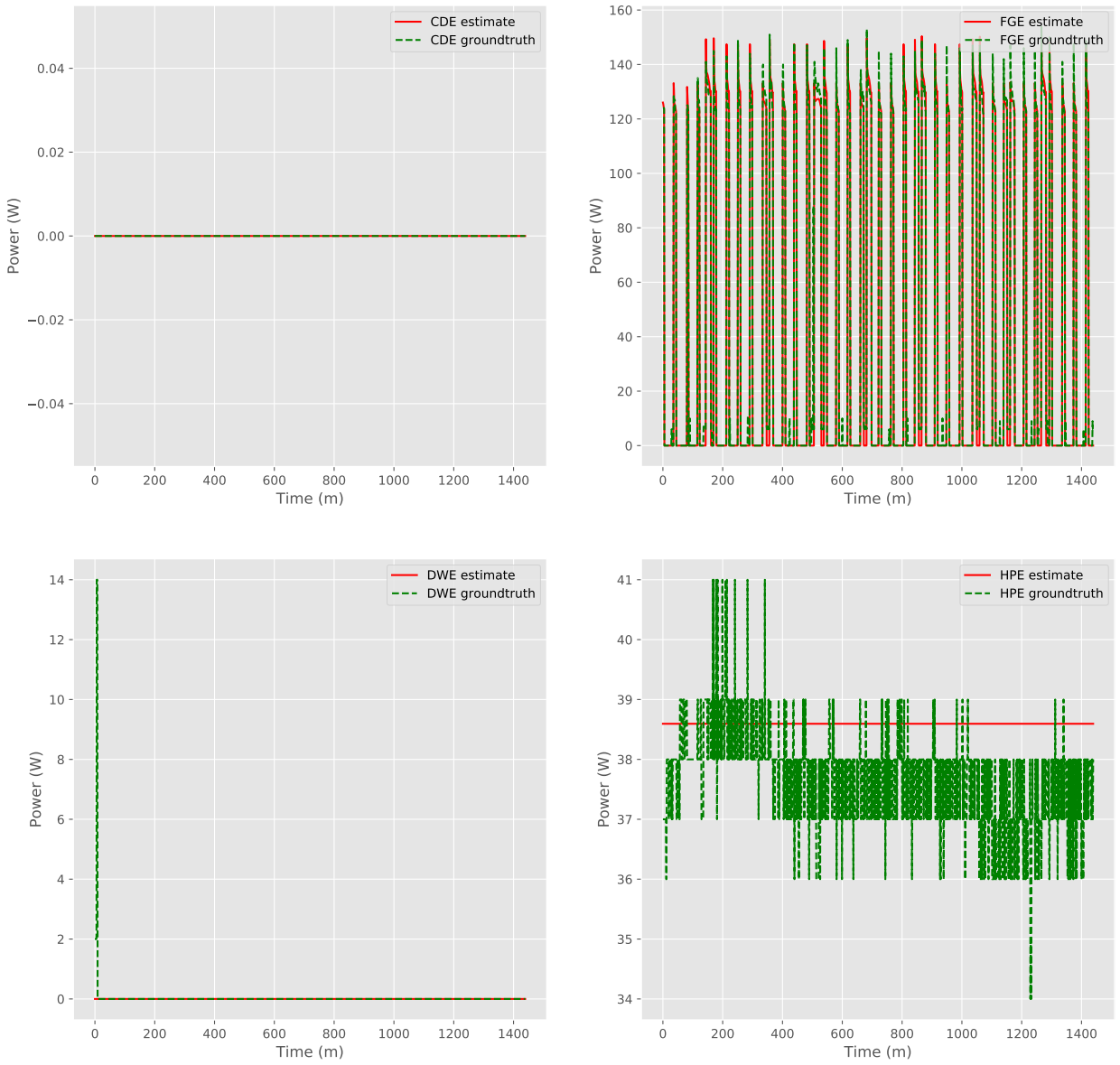

Figure A.1: Sample energy disaggregation results obtained by incorporating deep neural networks for device modeling and switching times prediction in the framework presented in chapter 3. Results are shown for four devices in red dashed lines with the groundtruth in green dashed lines. The four devices (left to right, top to bottom) are Clothes Dryer, Fridge, Dishwasher and Heat Pump. Results are shown for a day of data.

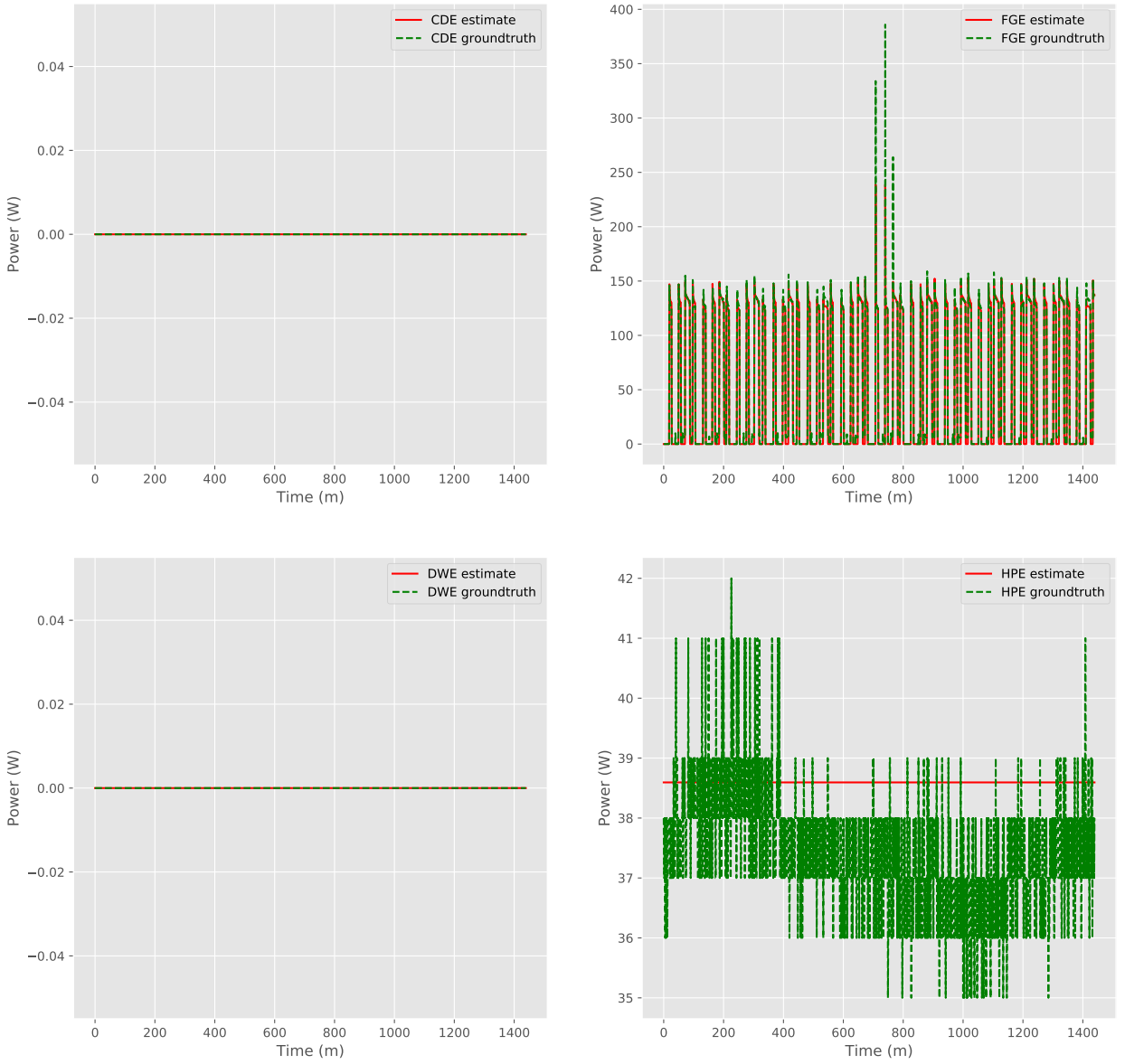

Figure A.2: Sample energy disaggregation results obtained by incorporating deep neural networks for device modeling and switching times prediction in the framework presented in chapter 3. Results are shown for four devices in red dashed lines with the groundtruth in green dashed lines. The four devices (left to right, top to bottom) are Clothes Dryer, Fridge, Dishwasher and Heat Pump. Results are shown for a day of data.

Appendix B

Supplementary Results for Chapter 4

B.1 Addition Results and Implementation Details

In order to simplify the presentation of the material we have excluded some more discussion on results and implementation details of the algorithm presented in chapter 4. We now provide these in the supplementary material. We also provide more implementation details and descriptions of the metrics used.

Metrics

We have made use of the F-score as recommended in [101] to provide a quantitative evaluation of our work. Given a groundtruth shape S and an estimate shape $\hat{S}$, as well as samples of points on their surfaces $\partial\Omega_S$ and $\partial\Omega_{\hat{S}}$, [1] the F-score is the harmonic mean of precision and recall metrics. Precision and recall are defined as follows:

To compute the precision, for each point y in the estimate shape, we first compute its distance to the groundtruth as:

$$e_y = \min_{x \in \partial\Omega_S} ||y - x||_2.$$

The precision is then computed as the percentage of points within distances less than ϵ_p:

$$P(\epsilon_p) = \frac{100}{|\partial\Omega_{\hat{S}}|} \sum_{y \in \partial\Omega_{\hat{S}}} \mathbf{1}_{\{e_y < \epsilon_p\}}. \tag{B.1}$$

Recall is computed in the opposite direction. Again, the error is first computed for each point x on the groundtruth shape as:

$$e_x = \min_{y \in \partial\Omega_{\hat{S}}} ||x - y||_2, \tag{B.2}$$

[1] For simplicity of notation, we have retained the same notation as that used for the set of all points on the surface. To compute the F-score, we can only use a sampling of the set of points on the surface.

and recall as:

$$R(\epsilon_r) = \frac{100}{|\partial\Omega_S|} \sum_{x \in \partial\Omega_S} \mathbf{1}_{\{e_x < \epsilon_r\}}. \tag{B.3}$$

Finally, the F-score is computed as:

$$F(\epsilon) = \frac{2P(\epsilon)R(\epsilon)}{P(\epsilon) + R(\epsilon)} \tag{B.4}$$

The F-score ranges from 0 to 1, with higher values indicating a better match between the estimate and the groundtruth. The F-score is very sensitive to deviations in shape and thus makes an excellent metric. To obtain pointclouds from the meshes, we have sampled 3000 points from the surface of each mesh. We have also set $\epsilon = 0.05r$, where r is the radius of the bounding sphere of the shape S.

An alternative metric may have been computing the distance between the estimated parameters and the groundtruth parameters, however, we find that solutions that are close in 3D space may not be close in the parameter space. For example, we find that the learned shapes can have different centers from their groundtruth counterparts, as such a good result may have a different translation value than the groundtruth.

Additional Results

In this section, we provide additional results to support those presented in chapter 4.

Synthetic data

We perform additional experiments making use of synthetic data to observe the behavior of our algorithm under varying initialization conditions. Following the same procedure described in section 4.4, we conduct two additional experiments that vary the initialization parameters as follows:

Table B.1: Parameter ranges for the difference between initialization points and groundtruth values.

Scenario	Δs	$\Delta\psi$	$\Delta\rho$	$\Delta\theta$	$\|\|\Delta t\|\|(m)$
Known rotation axis	$[0, 0.5)$	N/A	N/A	$[-\frac{2\pi}{9}, \frac{2\pi}{9})$	$[0, 0.20)$
Unknown rotation axis	$[0, 0.5)$	$[-\frac{\pi}{36}, \frac{\pi}{36})$	$[-\frac{\pi}{36}, \frac{\pi}{36})$	$[-\frac{2\pi}{9}, \frac{2\pi}{9})$	$[0, 0.20)$

In the first scenario in Table B.1, we assume a known axis a rotation, but increase the range of values that the initialization parameters can take on compared to those presented in chapter 4. In the second scenario, we relax the assumption of a known axis of rotation and introduce some uncertainty. We retain the same increase in the ranges of the other

parameters. The purpose of this experiment is to see how the performance degrades when we introduce more uncertainty.

As shown in Fig. B.1, we observe that both scenarios perform worse than those presented in chapter 4. We also observe that for most of the experiments the median F-score is still greater than 0.8. The difference here is that we observe significantly more F-scores in the $[0.5, 0.7]$ range.

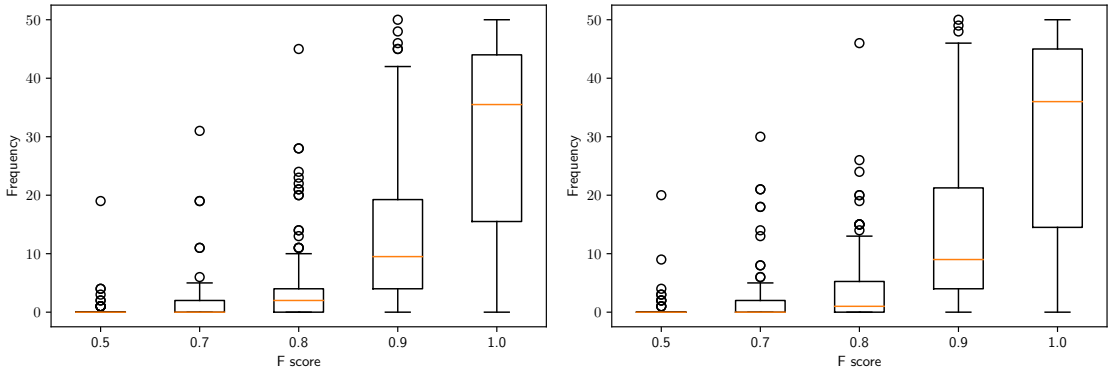

Figure B.1: Box plot of F@5% from experiments described in Table 4.1. **Left:** Scenario with a known axis of rotation; **Right:** Scenario with an unknown axis of rotation. The median for each bin is shown as an orange line in the box, with outliers shown as circles. Each box is placed on the right edge of its corresponding bin, the left edge of each bin is the right edge of the preceding bin. We still observe most median F-scores greater than 0.80, although we now see more cases of F-scores in the range $[0.5, 0.7]$.

Real Data

Figures B.2 - B.7 provide qualitative results from all the experiments carried out on the Redwood dataset [19] detailed in section 4.4. We have displayed comparisons to the benchmark making use of PointNet [81] and a variant of ICP [14] described in chapter 4. We report a similar level of performance as that presented in chapter 4. Our algorithm consistently provides qualitatively better results with fewer shape mismatch and misalignments. The figures also indicate that our algorithm sometimes struggles with fine details and star shaped legs. This is partly due to poor convergence of the SDF model during training and also because in reality the star-shaped legs are free to rotate separately from the main body of the chair. These additional degrees of freedom are not well captured by our assumptions.

Implementation Details

SDF Sampling

For experiments using synthetic data (section 4.4), we follow the same sampling procedure described in [76]. For experiments using the Redwood dataset [19] (section 4.4), we first preprocess the data by manually segmenting the floor and main object in each scene. We then make use of the plane fitting algorithm provided by [119] to segment the main object and the floor. We also obtain the axis of rotation in this fashion as the normal to the floor plane. Furthermore, we make use of the hidden point removal algorithm described in [52] and implemented in [119] to remove artifacts. Other artifacts and noisy points are also removed using the statistical outlier removal algorithm provided by [119].

Finally, for each point in the preprocessed mesh, we sample new points at a distance of ± 0.01m in the direction of the normal to the surface at each point. We estimate the SDF at these new points as ± 0.01 respectively. To obtain SDF samples representing the freespace, we take each point on the mesh and randomly and uniformly sample a new point between 0.07m and 0.20m away from the point on the mesh, in the direction of the outward pointing normal. We compute the SDF at these new points by finding the approximate distance to the closest point on the surface of the mesh. The sign is then determined by the whether each new point is in the direction of the outward pointing normal at its nearest neighbor on the mesh. We use a KDTree and the set of vertices on the densely sampled meshes to approximate the nearest neighbors. We obtain 25000 freespace SDF samples in this manner.

The SDF samples used during optimization are composed of the freespace SDF as well as the SDF sampled at distances ± 0.01m from points on the mesh.

Optimization Parameters

To solve the proposed optimization problem, we employ Adam [58] as implemented in [78]. We follow a stochastic gradient-descent approach and make use of 8000 SDF samples at each iteration. The learning rate is set to 0.05 for all parameters except for those assumed to be known. The learning rate is decreased by a factor of 5 every 400 iterations. We solve the optimization problem for 800 iterations.

Real data benchmarks

We report some implementation details for the benchmarks used for experiments on the Redwood dataset [19] presented in chapter 4.

Harris3D + SHOT

For this benchmark we make use of Harris3D keypoint detector and the SHOT[108] descriptor as implemented in [87].

OURCVFH + ICP

For this benchmark, we make use of OURCVFH [5] as implemented in [87] as a shape descriptor. We use the same subset of ShapeNet [17] chair models used to train our SDF model and compute a shape descriptor for each chair model in the subset. During test time we compute a OURCVFH[5] descriptor for the test model and use it to find the nearest neighbor from the subset of ShapeNet[17] chair models.

For registration, we first obtain a rough approximation of the optimal transform parameters. We estimate the scale as the radius of the bounding sphere for the test shape divided by the radius of the bounding sphere of its closest match from the subset of ShapeNet[17] chairs. We estimate the translation as the difference between the centers of their bounding spheres. Since we know the axis of rotation, we estimate the rotation by searching the space of rotation angles in increments of 20°. We select the rotation angle the minimizes the maximum difference between points on the test shape and their nearest neighbor on the closest matching shape.

We refine this estimate using an algorithm inspired by the Iterative Closest Point (ICP) algorithm [14]. We first sample the test shape and its closest matching shape uniformly to obtain sets of points representing each shape. Then for each iteration of the ICP algorithm, we find correspondences by solving an optimal assignment problem. We then estimate the optimal similarity transformation given the optimal correspondences. The optimal assignment problem gives us a one-to-one mapping between correspondences. This prevents us from obtaining a degenerate solution to the registration problem where the scale becomes almost zero. We make use of [113] to solve the optimal assignment problem and [111] as implemented in [119] to solve the registration problem.

PointNet + ICP

For this benchmark, we make use of the PointNet [81] classifier as implemented in [49]. We train the classifier to predict the id of the closest ShapeNet [17] model from our subset of chair models given an input pointcloud. During training, for each model we randomly sample 1024 point from the mesh and feed it to classifier for prediction. We add random noise to each of the points, and the set of points are re-sampled each epoch. We run it for 60 epochs after which the training loss does not significantly improve. We validate the model by making use of 30 models from the Redwood dataset [19]. We do not train on the Redwood dataset [19] models since they provide only a small number of models. Instead, we train on mesh models provided by our subset of ShapeNet [17] chair mesh models. During test time we sample 1024 points from the test shape and use it for predicting the closest matching model.

Registration is performed using the same algorithm described in the subsection above.

B.2 Derivation of Equation 4.7

The first half of eq. 4.7, was already derived earlier. The second half is highlighted in red below:

$$\min_{\substack{s,\psi,\rho,\\ \theta,t,z}} \sum_{(x_i,\phi_i)\in\chi_S} \left| s\hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(x_i-t)}{s}, z\right) - \phi_i \right| + \lambda \left| \hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(x_i-t)}{s}, z\right) - \hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(g(x_i)-t)}{s}, z\right) \right|.$$

We first note that the second half of this equation is a penalty term derived from the constraint that:

$$\hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(x_i-t)}{s}, z\right) = \hat{f}\left(\frac{[R(\psi,\rho,\theta)]^{-1}(g(x_i)-t)}{s}, z\right).$$

which is in turn obtained from the statement that:

$$\Phi(g(x), \Omega_{S'}) = \Phi(x, \Omega_S) \ \forall x \in \mathbb{R}^3$$

and that

$$\Omega_{S'} = g(\Omega_S) = \Omega_S. \tag{B.5}$$

However, the statement (B.5) is trivially true since g is in the symmetry group of Ω_S.

In essence, the second part of the equation (4.7) depends on showing that given shapes S, S' containing the closed sets of points $\Omega_S, \Omega_{S'} \subset \mathbb{R}^3$ respectively, and a reflective symmetry g such that $\Omega_{S'} = g(\Omega_S)$ then $\Phi(g(x), \Omega_{S'}) = \Phi(x, \Omega_S) \ \forall x \in \mathbb{R}^3$. As before, we assume that we deal with closed 3D shapes with a well defined notion of interior and boundary points.

From the definition of the scalar function Φ as:

$$\Phi(x, \Omega_S) = \begin{cases} -\min\limits_{y\in\partial\Omega_S} ||x-y||_2 \ , \text{ if } x \in \Omega_S \\ \min\limits_{y\in\partial\Omega_S} ||x-y||_2 \ , \text{ if } x \in \Omega_S^c \end{cases},$$

we see that we need to show two things. That

1. $\text{sgn}(\Phi(x, \Omega_S)) = \text{sgn}(\Phi(g(x), \Omega_{S'}))$
2. $|\Phi(x, \Omega_S)| = |\Phi(g(x), \Omega_{S'})|$

The main idea of the proof of these statements is as follows:

We note that since $\text{sgn}(\Phi(x, \Omega_S))$ is determined by whether or not $x \in \Omega_S$, the sign is preserved if g is a function that maps interior (boundary/ points outside the shape) points in the domain to interior (boundary/points outside the shape) points in the image and is also injective.

The magnitude of $(\Phi(x, \Omega_S))$ is determined by the distance of x to points on the boundary of Ω_S, thus the magnitude is preserved if g is a distance preserving function.

To show these statement, we will need to show the following two lemmas presented and proved in [103, 102], but also shown here for completeness:

Lemma B.2.1. *Let X, Y be topological spaces, let $f : X \to Y$ be a homeomorphism, and let $A \subseteq X$, then $f(int(A)) = int(f(A))$.*

***Proof**: Let $x \in int(A)$. Then there exists an open neighborhood $U \subseteq X$ such that $x \in U \subseteq A$. Consequently, $f(x) \in f(U) \subseteq f(A)$.*
Since f is a homeomorphism, $f(U) \subseteq Y$ is also open. Hence, we have an open neighbourhood $f(U) \subseteq f(A)$ in Y at $f(x)$, making $f(x) \in int(f(A))$.
We have just shown that $x \in int(A) \implies f(x) \in int(f(A))$. The converse statement is also true and the proof is identical to the one shown above. □

Lemma B.2.2. *Let X, Y be topological spaces, let $f : X \to Y$ be a homeomorphism, and let $A \subseteq X$, then $f(\partial A) = \partial(f(A))$.*

***Proof**: Let $x \in f(\partial A)$, then for any open ball U at x, there exists an open ball $f^{-1}(U)$ in X containing $f^{-1}(x)$, since f is a homeomorphism.*
Since $f^{-1}(x) \in \partial A$, then there exists points $a, b \in f^{-1}(U)$, $a \neq b \neq f^{-1}(x)$ such that $a \in A \cap X$ and $b \in A^c \cap X$.
Consequently, there exists $f(a), f(b) \in U$ such that $f(a) \in f(A) \cap Y$ and $f(b) \in f(A^c) \cap Y$. However, since f is injective, $f(A^c) \subseteq f(A)^c$. Therefore, there exists $f(a), f(b) \in U$ such that $f(a) \in f(A) \cap Y$ and $f(b) \in f(A)^c \cap Y$. Hence, $x \in \partial f(A)$. Note that $f(a) \neq f(b) \neq x$, since f is injective. The proof of the converse statement is similar to that shown above. □

Putting together Lemma B.2.1 and Lemma B.2.2, we get the following statement:

Remark B.2.1. *Let X, Y be topological spaces, let $f : X \to Y$ be a homeomorphism, and let $A \subseteq X$ and closed, then $x \in A \iff f(x) \in f(A)$. We also note that the contrapositive is also true.*

Since g is a Euclidean transform, g is also a homeomorphism. Hence the statements above and the definition of Φ prove that $\text{sgn}(\Phi(x, \Omega_S)) = \text{sgn}(\Phi(g(x), \Omega_{S'}))$ □

To complete the proof we note the following Statement:

Lemma B.2.3. $|\Phi(x, \Omega_S)| = |\Phi(g(x), \Omega_{S'})|$

***Proof**: This is true since, g is a Euclidean transform (distance preserving) and g is a homeomorphism.*
Using the Lemmas proved above, we know that $g(\partial\Omega_S) = \partial g(\Omega_S) = \partial\Omega'_S$.
Consequently, $|\Phi(g(x), \Omega_{S'})| = \min_{y \in \partial\Omega_S} ||g(x) - g(y)||_2$. But, by the distance preserving property of g, $\forall y \in \partial\Omega_S$, $||g(x) - g(y)||_2 = ||x - y||_2$.

Hence, $|\Phi(g(x), \Omega_{S'})| = \min_{\bar{y} \in \partial\Omega_{S'}} ||g(x) - \bar{y}||_2 = \min_{y \in \partial\Omega_S} ||g(x) - g(y)||_2 = \min_{y \in \partial\Omega_S} ||x - y||_2 = |\Phi(x, \Omega_S)|$ □

Finally, $\text{sgn}(\Phi(x, \Omega_S)) = \text{sgn}(\Phi(g(x), \Omega_{S'}))$ and $|\Phi(x, \Omega_S)| = |\Phi(g(x), \Omega_{S'})| \implies \Phi(x, \Omega_S) = \Phi(g(x), \Omega_{S'})$ since Φ is a scalar valued function.

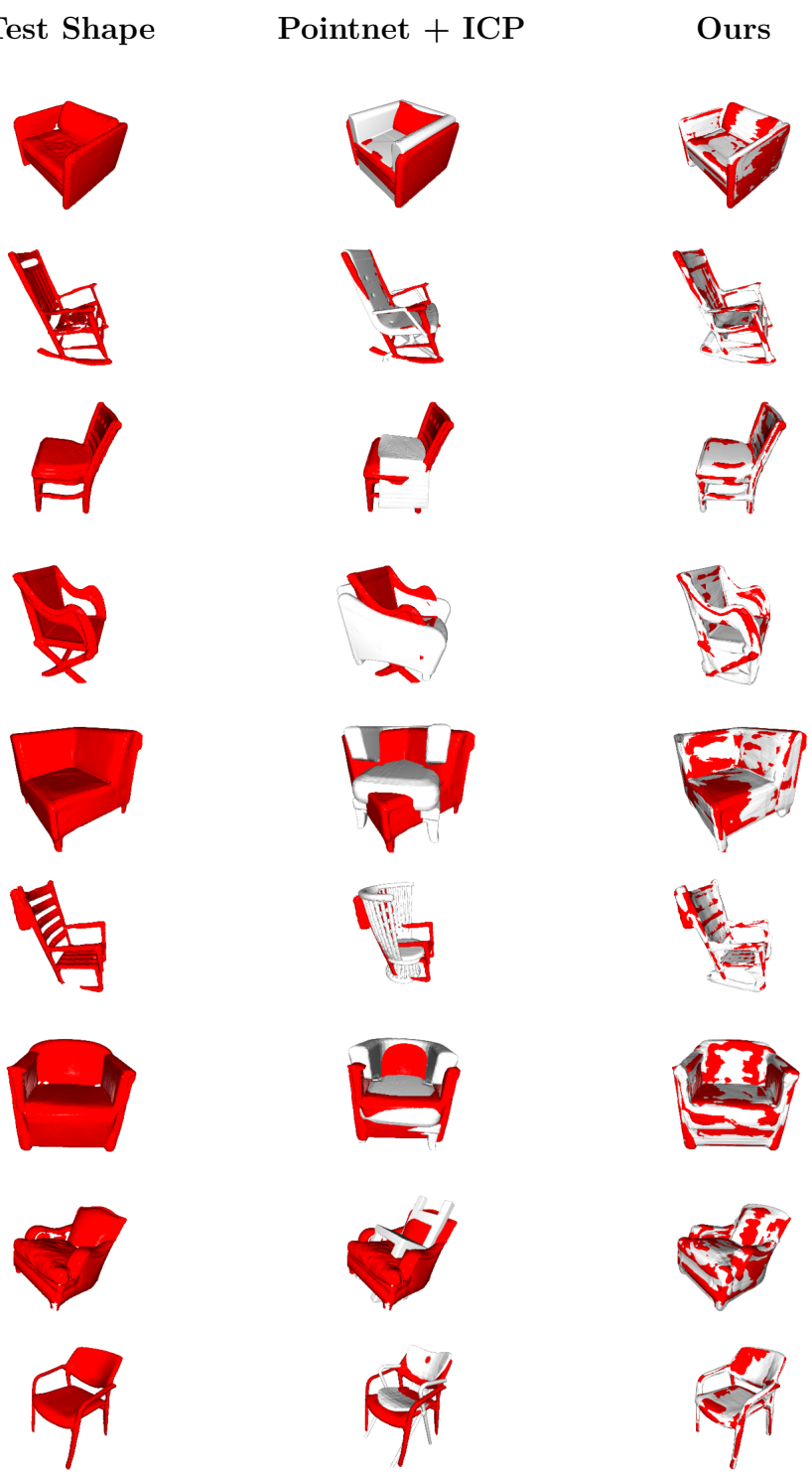

Figure B.2: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [81] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in chapter 4.

Figure B.3: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [81] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in chapter 4.

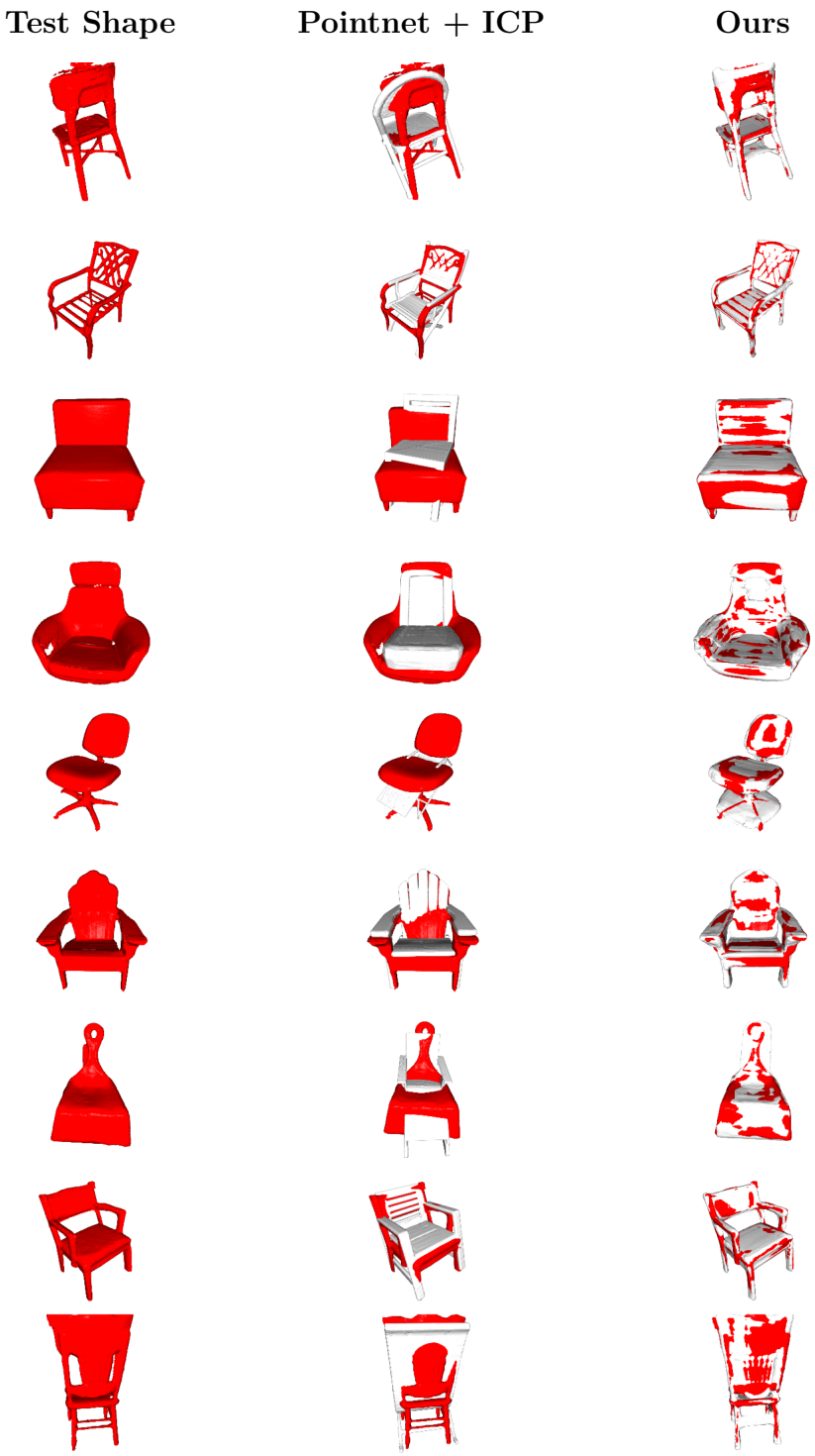

Figure B.4: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [81] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in chapter 4.

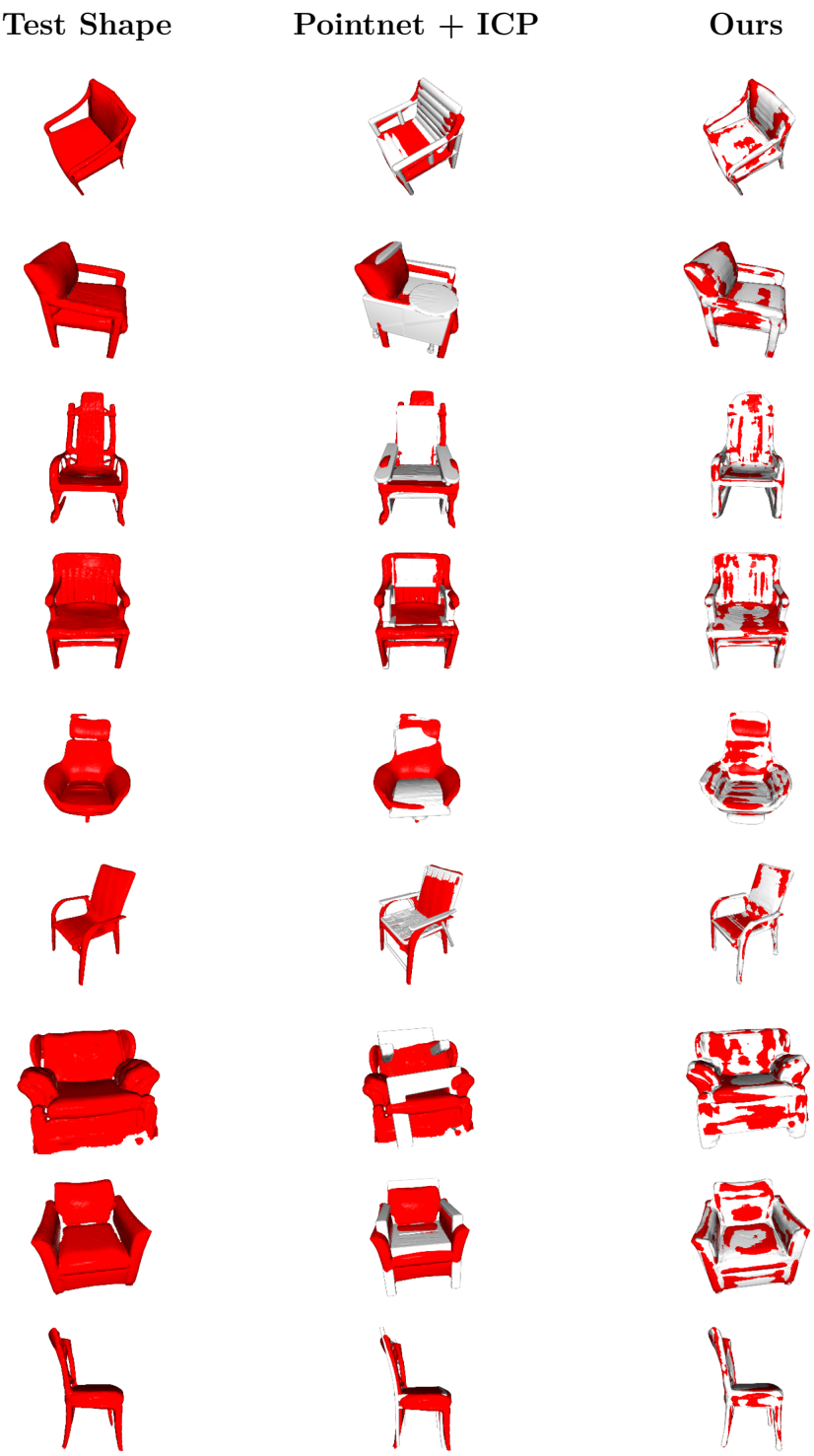

Figure B.5: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [81] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in chapter 4.

Figure B.6: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [81] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in chapter 4.

Figure B.7: Additional qualitative result on Redwood dataset. Left to Right, test shape, results using PointNet [81] and ICP, our proposed method. We have shown each method's result in gray overlaid with the input mesh in red. Our algorithm provides significant qualitative improvement on par with those displayed in chapter 4.