AN ABSTRACT OF THE DISSERTATION OF

Austin Nicolai for the degree of Doctor of Philosophy in Robotics presented on September 11, 2019.

Title: Augmented Deep Learning Techniques for Robotic State Estimation

Abstract approved: ______________________________

Geoffrey A. Hollinger

While robotic systems may have once been relegated to structured environments and automation style tasks, in recent years these boundaries have begun to erode. As robots begin to operate in largely unstructured environments, it becomes more difficult for them to effectively interpret their surroundings. As sensor technology improves, the amount of data these robots must utilize can quickly become intractable. Additional challenges include environmental noise, dynamic obstacles, and inherent sensor non-linearities. Deep learning techniques have emerged as a way to efficiently deal with these challenges. While end-to-end deep learning can be convenient, challenges such as validation and training requirements can be prohibitive to its use.

In order to address these issues, we propose augmenting the power of deep learning techniques with tools such as optimization methods, physics based models, and human expertise. In this work, we present a principled framework for approaching a problem that allows a user to identify the types of augmentation methods and deep learning techniques best suited to their problem. To validate our framework, we consider three different domains: LIDAR based odometry estimation, hybrid soft robotic control, and sonar based underwater mapping.

First, we investigate LIDAR based odometry estimation which can be characterized with both high data precision and availability; ideal for augmenting with optimization methods. We propose using denoising autoencoders (DAEs) to address the challenges presented by modern LIDARs. Our proposed approach is comprised of two stages: a novel pre-processing stage for robust feature identification and a scan matching stage for motion estimation. Using real-world data from the University of Michigan North Campus long-term vision and LIDAR dataset (NCLT dataset) as well as the KITTI dataset, we show that our approach generalizes across domains; is capable of reducing the per-estimate error of standard ICP methods on average by 25.5% for the translational component and 57.53% for the rotational component; and is capable of reducing the computation time of state-of-the-art ICP methods by a factor of 7.94 on average while achieving competitive performance.

Next, we consider hybrid soft robotic control which has lower data precision due to real-world noise (e.g., friction and manufacturing imperfections). Here, augmenting with model based methods is more appropriate. We present a novel approach for modeling, and classifying between, the system load states introduced when constructing staged soft arm configurations. Our proposed approach is comprised of two stages: an LSTM calibration routine used to identify the current load state and a control input generation step that combines a generalized quasistatic model with the learned load model.

We show our method is capable of classifying between different arm configurations at a rate greater than 95%. Additionally, our method is capable of reducing the end-effector error of quasistatic model only control to within 1 cm of our controller baseline.

Finally, we examine sonar based underwater mapping. Here, data is so noisy that augmenting with human experts and incorporating some global context is required. We develop a novel framework that enables the real-time 3D reconstruction of underwater environments using features from 2D sonar images. In our approach, a convolutional neural network (CNN) analyzes sonar imagery in real-time and only proposes a small subset of high-quality frames to the human expert for feature annotation. We demonstrate that our approach provides real-time reconstruction capability without loss in classification performance on datasets captured onboard our underwater vehicle while operating in a variety of environments.

©Copyright by Austin Nicolai
September 11, 2019
All Rights Reserved

Augmented Deep Learning Techniques for Robotic State Estimation

by
Austin Nicolai

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented September 11, 2019
Commencement June 2020

Doctor of Philosophy dissertation of Austin Nicolai presented on September 11, 2019.

APPROVED:

Major Professor, representing Robotics

Associate Dean for Graduate Programs, College of Engineering

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Austin Nicolai, Author

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Geoffrey Hollinger, for his continued support and mentorship during my time at OSU. I would also like to thank Drs. William Smart, Kagan Tumer, Fuxin Li, and Jihye Park for their suggestions and advice on this work as members of my committee.

Next, I would like to thank all of my collaborators with whom I have had the pleasure of working with for their contributions to the work presented in this dissertation: Robert DeBortoli, Scott Chow, and Gina Olson. I would also like to thank the many members of the Robotics Decision Making Laboratory and Personal Robotics Group for their feedback, idea-vetting, and support throughout this process.

Finally, I would like to thank my family for their constant support. Thank you to my parents, Gisele and Steven, for always encouraging me and helping to keep me motivated. Thank you to my brother, Ryan, and sister, Erin, for reminding me to never take things too seriously. Without you, none of this would be possible.

This work has been funded in part by NASA (NNX14AI10G), DWFritz Automation, the Oregon Metals Initiative, U.S. Department of Energy (DE-EE-0006816), and National Science Foundation (IIS-1734627).

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

In today's society, robotic systems are becoming more and more prevalent. While these systems may have once been relegated to structured environments and automation style tasks, in recent years these boundaries have begun to erode. Now, the promise of these systems is beginning to reach into all aspects of life whether it is on a personal or global scale. Applications ranging from autonomous vehicles and delivery to disaster relief all promise to have a large impact on humanity. As these applications move into more unstructured environments, with increasing human contact, the ability of a robot to efficiently perceive and act on information from its environment becomes increasingly important.

As these robots begin to operate in largely unstructured environments, it becomes more difficult for the robot to effectively interpret its surroundings. While this may seem unintuitive at first, especially with modern sensors becoming both more capable and cheaper, there are many reasons for this. As sensor technology improves, the amount of data that these robots must utilize can quickly become intractable; especially when many sensor modalities are present on a single robot. Additional challenges include environmental noise, dynamic obstacles, and inherent sensor non-linearities (e.g., sonar, soft sensors).

Deep learning techniques have emerged as a way to efficiently deal with these challenges. Beginning with perception tasks such as object recognition and classification, deep learning has shown to be highly successful [51, 52, 95]. In these domains, deep learning has allowed training datasets to grow from an order of thousands (e.g., LabelMe [80]), to tens of thousands (e.g., CIFAR-10/100 [50], NORB [54]), and even millions of training examples (e.g., ImageNet [19]).

As deep learning techniques have started to enter the robotics space, many problems have been approached in an end-to-end fashion. In the perception domain, end-to-end deep learning has been proposed to predict odometry from sequences of stereo images [49] as well as to detect and track liquids in visual scenes using single and multiple frame inputs [81]. Other approaches have used LIDAR data for vehicle detection [57], to estimate the positions and classes of objects in the environment [72, 73], as well as to perform landing zone detection for autonomous helicopters [64]. In the grasping domain, raw RGB-D input has been used to evaluate candidate grasps and propose the top-ranked candidate for Baxter and PR2 robots [56]. Additionally, end-to-end deep learning has been used to model the complex dynamics of a helicopter as a high dimensional regression problem [75].

While end-to-end deep learning can be convenient in terms of system architecture (i.e., mapping raw data to a desired output), there are drawbacks. Such systems are difficult to validate and when a problem arises, it can be difficult to diagnose where in the pipeline the issue occurred. Additionally, training these end-to-end systems requires extremely large amounts of data and computation time. In order to address these issues, we propose augmenting the power of deep learning techniques with tools such as optimization methods, physics based models, and human expertise.

In this work, we propose a principled framework for approaching a novel problem that allows the user to identify the types of augmentation methods and deep learning techniques best suited to their problem. Our framework is composed of two steps that examine the structure of the problem. In the first step, the data precision and availability is analyzed to determine the most appropriate type of augmentation method. The second step analyzes both the spatial and temporal locality of the feature space to suggest the appropriate deep learning techniques. A visualization of these spaces can be seen in Fig. 1.1 and Fig. 1.2, respectively.

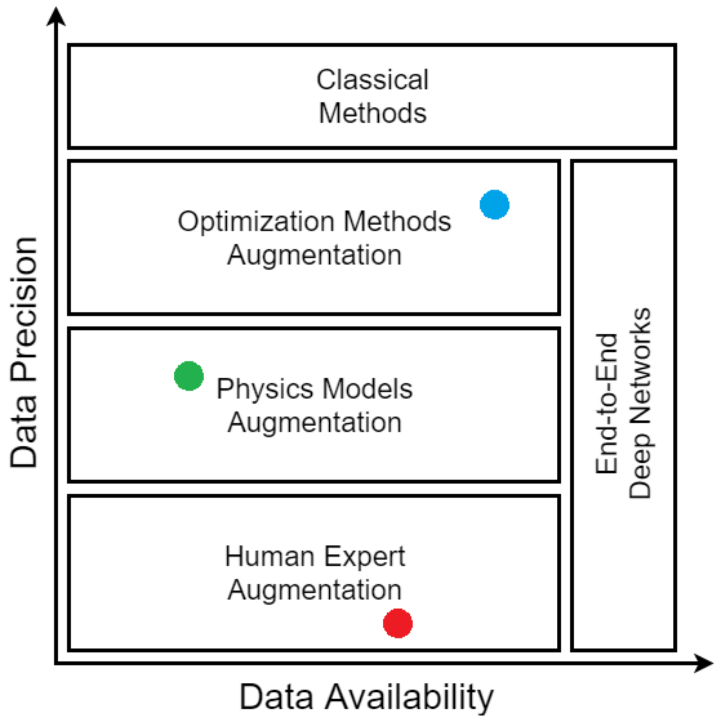

Classical Methods
Optimization Methods Augmentation
Physics Models Augmentation
Human Expert Augmentation
End-to-End Deep Networks
Data Precision
Data Availability

Figure 1.1: A visualization of the data space and various augmentation methods. The colored dots represent the three domains investigated in this work. Blue represents LIDAR based odometry estimation, green represents hybrid soft robotic control, and red represents sonar based underwater mapping.

As seen in Fig. 1.1, the entire space is not characterized by augmentation methods.

When the problem is characterized by extremely precise data, deep learning techniques are not generally required. While these problems are less frequent, classical methods tend to work well in them (e.g., noiseless sensors in simulation). When the problem is characterized by extreme data availability, end-to-end deep learning is an option. In these types of problems, the vast amount of data available allows for the learning space to be fully explored (e.g., hundreds of instances of hardware collecting data for weeks or months on end). However, due to cost and time constraints, this is not feasible for many problems.

If the problem has precise data (e.g., LIDAR), optimization methods can be used. The more precise the data, the better these methods will work since they will be able to more easily converge. However, these precise modalities are often accompanied by large quantities of data (i.e., individual LIDAR scans can contain tens of thousands of individual returns). Here, deep learning can be leveraged to reduce the input data to only the most important. As the data precision degrades, physics based model approaches may be more suitable. In many robotic systems, data precision can be compromised by physical limitations (e.g., manufacturing defects) as well as complex system dynamics (e.g., hysteresis). When this occurs, straightforward models can be combined with deep learning to more accurately model the full complexity of the system. In some cases, data precision degrades to a point where it is nearly indistinguishable from noise. As this limit is approached, human experts can be utilized for their domain knowledge to identify key elements from the data. While human experts can be an invaluable resource, their time and energy is finite. In these scenarios, deep learning can be used to reduce the amount of data the human has to process to only the most useful and bare minimum for the task at hand.

It is important to note that the data precision and availability are not the only factors to consider when augmenting deep learning techniques. When using the proposed augmentation methods, various other factors should be considered. In order to use optimization methods, the problem must be formulated in such a way that there is a clear objective function. In addition, it is important to consider the properties of the objective function (e.g., convexity). Similarly, in order to use a model based approach, a model (of some quality) must exist for the system of interest. Another consideration is whether or not data representing the additional system complexities can be obtained. For example, if attempting to model human behavior for human-robot interaction tasks, it may not be feasible or ethical to collect specific training data for certain complex behaviors. Finally, in order to utilize a human expert, one must exist for the domain of interest; this may not be the case for many data modalities (e.g., raw binary data). Additionally, the domain must be one in which an expert could intervene. For many long-term autonomous aerial and underwater deployments, direct communication with the vehicle may not be possible.

Fig. 1.2 visualizes the types of deep learning techniques we consider. On the vertical axis, spatial locality ranges from no correlation (bottom) to purely global correlation (top). On the horizontal axis, temporal locality ranges from no correlation (left) to highly correlated (right). When the spatial features are globally correlated, with no local correlation, classical approaches such as discrete cosine transforms may be appropriate. If the features have a mix of global and local correlation, atrous convolution can be leveraged. This allows the network to take advantage of semi-global context, similar to

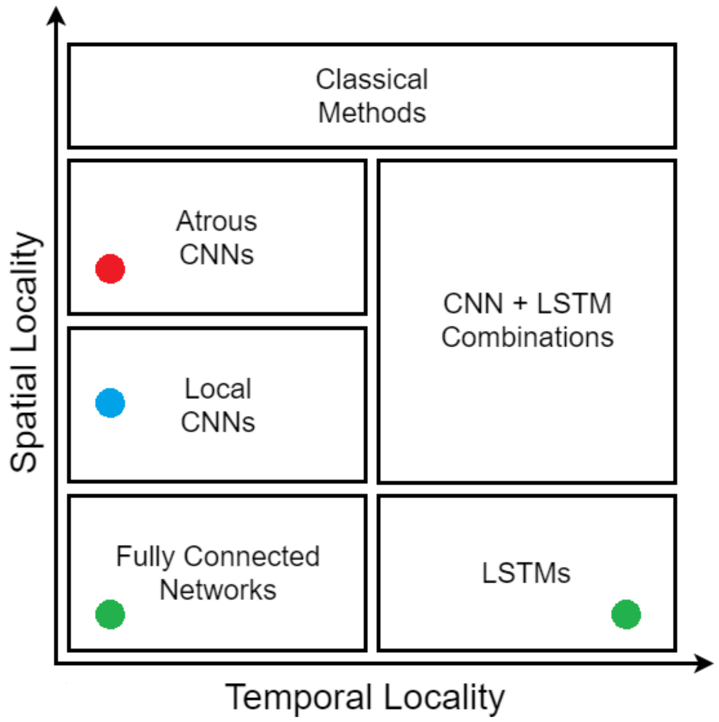


Figure 1.2: A visualization of the feature space and various deep learning techniques. The colored dots represent the three domains investigated in this work. Blue represents LIDAR based odometry estimation, green represents hybrid soft robotic control, and red represents sonar based underwater mapping. For hybrid soft robotic control, multiple network structures are used to take advantage of the different input spaces across the approach. This is further discussed in Section 4.2.

convolution and pooling layer pairs, without the loss of feature information introduced by the pooling operator. When the features are only locally correlated, standard local convolution is appropriate. Finally, if there is no spatial correlation, more simple fully connected networks can be used. If there is a temporal aspect to the features, LSTMs can be used on their own or in conjunction with convolution layers. While we do not use the combination of LSTMs and convolution layers in this work, this technique would be useful for data such as video sequences where features exhibit both a spatial and temporal component.

In order to validate our framework, we consider three different domains, each with a different task. For each domain, we follow our framework to determine the augmentation method and deep learning techniques best suited to the challenges presented. These domains are represented by the colored dots in Fig. 1.1 and Fig. 1.2. More specifically, the three primary contributions are as follows:

1. A novel pre-processing step for LIDAR based odometry estimation that both removes outlier returns and reduces the size of the points clouds to align. This domain is an example of high data precision and availability, with local features.

2. A scalable hybrid controller for reconfigurable staged soft arms that can switch between different arm configurations on the fly, requiring no addition learning or parameter fitting. In this domain, data is less precise due to real-world noise (e.g., friction and manufacturing imperfections). The different stages of this approach include both temporal and non-temporal features.

3. A method to identify informative sonar frames enabling real-time human-in-the-loop capabilities for underwater mapping tasks. The data precision in this domain is low enough that features are nearly indistinguishable from local noise. Due to this, some global context is required.

Chapter 2: Background

In this work, we leverage existing literature from many fields. In Section 2.1, we introduce deep learning and discuss common structures, training methods, and practical considerations. Next, we detail background, challenges, and approaches in the three domains considered in this work. LIDAR based odometry estimation is discussed in Section 2.2, hybrid soft robotic control in Section 2.3, and finally sonar based underwater mapping is in Section 2.4.

2.1 Deep Learning

In recent years, a new trend known as *deep learning* has emerged in the field of machine learning. Often, *deep learning* refers to "deep" multi-layer neural networks. However, the term can also be applied to other types of multi-layered, trainable architectures. Until recently, it seemed that training these deep networks was not feasible. With prior methods, the early layers in a deep network were unable to learn useful information. In fact, deep networks had been shown to generally perform the same, or worse, than shallow neural networks (i.e., one or two layers) [87].

It was not until 2006 that the difficulties in training these deep networks were addressed. At this time, the research groups of Hinton, LeCun, and Bengio proposed methods that allowed the early layers in deep networks to learn useful features [29, 28, 2, 76].

When comparing these deep networks to standard neural networks, there are both similarities and differences. Both types of networks are built up from layers of interconnected neurons. These neurons can use various types of non-linear activation functions and regularization parameters (Section 2.1.1). A key difference between the two network types are the layer structures employed. While standard neural networks typically use dense layers (i.e., all neurons fully connected pairwise between layers), deep networks leverage multiple layer structures including convolutional and recurrent layers (Section 2.1.2).

2.1.1 Layer Parameters

Common non-linear activation functions used in deep networks include the sigmoid, hyperbolic tangent, and Rectified Linear Unit (ReLU) [37]:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

$$tanh(x) = 2 \cdot \sigma(2x) - 1, \quad \sigma(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

$$ReLU(x) = max(0, x). \tag{2.3}$$

The sigmoid function is a standard activation function and bounds the output between 0 and 1. Alternatively, the hyperbolic tangent function bounds the output between -1 and 1 allowing for negative output values. In recent years, the ReLU has become popular due to its characteristics of sparsity ($x \leq 0$) and lower probability of a vanishing gradient ($x > 0$).

In order to combat the issue of overfitting, deep learning can employ a variety of regularization methods. Regularizations can be applied to a layer's weight terms, bias term, or both. Some of the standard regularizations that are commonly used include L1 and L2 regularization:

$$\lambda \, ||\boldsymbol{w}||_1 = \lambda \sum_{i=1}^{n} |w_i| \tag{2.4}$$

$$\lambda \, ||\boldsymbol{w}||_2^2 = \lambda \sum_{i=1}^{n} w_i^2 \tag{2.5}$$

where $\boldsymbol{w}$ represents the weight vector and λ represents a penalty term.

Additionally, max-norm constraints can be used to enforce an upper bound on the magnitude of the weight vectors for each neuron [85]:

$$||\boldsymbol{w}||_2 < c \tag{2.6}$$

where $\boldsymbol{w}$ represents the weight vector and a typical constraint value used is $c = 4$. Finally, dropout is a unique form of regularization that was presented by Srivastava et al. and has been shown to be extremely simple yet effective [86]:

$$w_i = w_i \cdot \text{Bernoulli}(p) \tag{2.7}$$

where w_i represents an individual weight and the Bernoulli random variable has probability p of being 1. This amounts to sampling a subset of each layer's output and using this subset as the input to the next layer. Note that at test time, dropout is turned off.

Outputs of a deep network can take the form of continuous variables (e.g., regression) or as probabilities (e.g., classification). Some examples of optimizers that can be used include standard gradient descent, RMSProp, Adadelta [101], and Adam [48]. RMSProp, Adadelta, and Adam are all per-parameter, adaptive optimization methods.

2.1.2 Layer Architectures

2.1.2.1 Fully Connected

Fully connected layers are connected the same way in deep networks as they are in standard neural networks. In these layers, neurons between two adjacent layers are fully pairwise connected. Neurons within a layer share no connections. These are rarely used in early layers of a deep learner due to the typically large input space; that is, intractable numbers of connections would be made.

2.1.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are biologically inspired. Early work by Hubel and Wiesel proposed an explanation for the way in which the visual cortex functions [35, 34]. They proposed that the visual cortex is comprised of cells which are sensitive to sub-regions of the visual field. These sub-regions are known as the *receptive field*. Aiming to emulate this behavior, several models were created including the NeoCognitron [21] and LeNet-5 [53].

In an attempt to replicate the concept of receptive fields, layers in a CNN are not fully connected. Instead, a sub-region (typically small; e.g., 3x3, 5x5) is defined and only the encompassed neurons are connected. Each sub-region, or filter, is then replicated across the entire input space. Each of the replicated units share the same weight vector and bias term. This allows for the same features to be detected at any region of the input space, as well as drastically reducing the number of free parameters being learned.

Structurally, in a CNN, convolutional layers are typically followed by *pooling layers*. Pooling layers are simple, and can be thought of as a way to intelligently down sample. Pooling layers partition the input into a set of non-overlapping sub-regions, each of which produces a single output. Pooling can take several forms including max and mean pooling. In max pooling, the output is simply the max value of the input sub-region, while mean pooling outputs the mean.

2.1.2.3 Recurrent Neural Networks

The idea behind recurrent networks (RNNs) is simple. In standard feed-forward networks, all inputs are treated as independent of one another. In reality, this is not always the case; RNNs attempt to address this. Recurrent layers do so by having a *memory* that retains information about calculations that have occurred. In practice, this memory is typically limited to a small number of time steps.

Long short-term memory (LSTM) networks are a type of RNN [30]. While RNNs are typically limited in their memory length, and may not learn terms that occur over the long-term, LSTMs are designed to avoid this issue. LSTMs store information outside of the normal flow of the network, in units called *gates*.

These gate units are able to make decisions about what information to store and when to allow reads and writes. Similar to a neural network, these gates act on the signals received. LSTMs maintain their own, internal, set of weights that are adjusted iteratively using gradient descent. That means that each cell learns when to allow data to enter, to leave, or to be deleted over time.

2.1.3 Training Methods

As previously mentioned, deep learning did not become tractable until 2006 when Hinton originally proposed an unsupervised, greedy layer-by-layer training approach for deep belief networks (DBNs) [28]. DBNs are multilayer generative models in which each layer encodes statistical dependencies of the layer below. Once each layer has been learned, the entire network can be trained as a whole to fine tune the weights. Later, autoencoders were proposed as a method to initialize deep network layers [2, 76].

An autoencoder is a mirrored neural network that has the same sized output as input. Each increasingly narrow hidden layer can be regarded as compressing the input into a reduced state space. Once the state space has been reduced to a desired size, the layers are reversed to reconstruct the input. Autoencoders are trained by attempting to minimize the error in this reconstruction.

One option for training autoencoders is to train a single layer at a time. These are known as Stacked Autoencoders (SAEs) [63, 94]. Once a layer has been learned, the next layer is added and the previous layer weights are locked (i.e., not updated anymore). SAEs have been shown to aid in learning more robust features.

Denoising autoencoders are another way to increase the robustness of learned features [94]. In these, the original training set is augmented by adding new training examples that are simply previous example with added noise. For the newly added examples, the loss is computed by comparing the output to the original, non-noisy training example. Additionally, pre-training the weights of convolutional layers of a larger network with an autoencoder has been shown to produce more robust filters [49].

2.1.4 Real-World Considerations

The typically large number of weights in a deep network leads to requiring large amounts of training data. These large amounts of data result in extremely long training times; times that would typically be intractable. However, with the recent advances in massively parallel GPU computing, these times are reduced to a tractable level [66].

In implementing deep networks, a variety of software frameworks are available. These frameworks allow the user to more quickly develop and train deep architectures by providing implementations of a variety of layers and training methods. One such framework is Caffe, developed at the University of California, Berkeley [39]. Another, much lower level framework, is Theano [88]. Theano gives the user very direct control over each layer, but can be tedious if this control is not needed. Both Lasagne and Keras offer wrappers to Theano, and provide many of the standard layers and training types that a user would desire [20, 15].

2.2 LIDAR Based Odometry Estimation

2.2.1 Classical Approaches

One simple method for performing odometry estimation is wheel based odometry. Wheel based odometry utilizes encoders on a robot's wheels, factoring wheel rotation into a forward kinematic model to estimate the motion of the robotic base. This process is referred to as *dead reckoning* and is known to result in a diverging estimate of a robot's position over time. In addition to wheel encoders, classical odometry measures often use a combination of accelerometers and gyroscopes, both of which are known to give noisy estimates.

A more robust estimation process is known as *scan matching* (e.g., [59, 97]), which often uses a technique known as Iterative Closest Point (ICP) [13] to iteratively minimize the difference between point clouds; in this case two laser scans. ICP based techniques are employed in many localization and mapping methods that rely on LIDAR data. In [5], Bosse and Zlot use ICP based techniques to estimate trajectories from data collected on a moving vehicle. They further employ their method to map a mine in New South Wales, Australia [105]. More recently, Zhang and Singh achieve low-drift odometry and mapping without loop closure by decomposing the problem into two simultaneously running algorithms [102, 103]. First they estimate odometry by extracting feature points from the data and performing scan matching in real-time (10 Hz). Mapping is then performed by fine-grained registration of entire point clouds in non real-time (1 Hz).

In [3], Bergstrom and Edlund use robust M-estimation techniques to reduce the effect of outlier points as well as comparing the ability of different criterion functions to do point set fits. Additionally, Luong et al. investigate methods to improve ICP techniques in sparse point clouds (particularly along one dimension), showing their method to be robust in the presence of noise [60]. Despite popular use and developments in efficiency, ICP can still be expensive to compute. Additionally, manually extracting features is undesirable.

2.2.2 Deep Learning Approaches

Currently, deep networks are successfully being applied to visual odometry (e.g., [10, 70]), a process that uses image features to estimate the ego-motion of a camera. One such method by Konda and Memisevic uses a CNN to estimate odometry from a sequence of stereo images [49]. In their work, odometry estimates are provided as a classification of discretized direction and velocity changes. Other examples include PoseNet [45] and PlaNet [98]. In PoseNet, Kendall et al. use CNNs to regress a 6 degree of freedom (DOF) camera pose from a single RGB image, showing success in both indoor and outdoor environments. In PlaNet, Weyand et al. localize photos to their geolocation by dividing the Earth's surface into geographic cells and treating the problem as a classification task. Most recently, new spatial transform modules have been proposed that allow for direct spatial manipulation of data within a network [36]. Handa et al. have extended these layers to 3D transformations, performing early experiments on visual odometry as a sanity check [27].

Despite these successes, deep networks have yet to see many applications to LIDAR data. One recent example uses deep networks to perform vehicle detection from projected LIDAR scans [57]. This demonstrates that meaningful features can be extracted from LIDAR data, despite the challenges presented by range and sparsity.

2.3 Hybrid Soft Robotic Control

2.3.1 Modeling Approaches

In soft robotics, kinematic modeling approaches commonly utilize the constant curvature approximation. An overview of these techniques are presented by Webster and Jones in [96]. In cases where actuator characteristics can be more easily measured (e.g., cable driven arms) more precise geometric models can instead be used [78, 79]. Controllers that incorporate sensory feedback have also been proposed. Such methods include both single camera [16] and multiple camera [9] controllers.

2.3.2 Learning Based Approaches

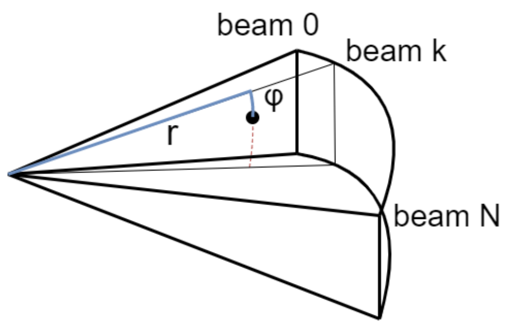
In recent years, many data-driven learning based approaches have been proposed for controlling soft robots. An early model-free implementation of a static controller was proposed by Giorelli et al. in which the inverse statics of a cable driven soft robot were directly learned with a neural network [26]. This approach was validated on physical hardware for a 2 DOF [25] and 3 DoF [24] soft manipulator and was shown to outperform the more complex model used for comparison. An approach learning the inverse kinematics of a soft robot was presented by Thurunthel et al. in [89, 90]. In this approach, the problem was formulated as a differential inverse kinematics problem using local mapping which allows for multiple solutions globally. More recently, Thurunthel et al. have proposed an approach that additionally incorporates end-effector feedback for learning the inverse kinematics [91]. They show that their algorithm can deal with stochasticity and exhibits adaptive behavior in an unstructured environment. Finally, Bruder et al. have proposed using Koopman operator theory to learn linear dynamical models for a non-linear soft robot arm [7]. The benefit to this method is that in the lifted linear state, the optimization problem is convex allowing for use with model predictive control techniques. While these data-driven methods perform well, they do not extend to generalizable arm configurations which is desirable for reconfigurable staged arm architectures.

In addition to solely data-driven methods, hybrid controllers that combine both model based and model free approaches have been proposed. Jiang et al. have proposed a hybrid approach that first uses a cost function to transform between task and configuration space [40]. Once a pose has been identified, a neural network is used to map curvature and arc length to control pressures. Another approach proposed by Reinhart et al. observes that errors in the constant curvature model reduce the accuracy of the inverse kinematic controller [77]. To reduce this error, they model the error with a neural network and show that their hybrid method is capable of reducing tracking error for their arm. In our work we use a similar hybrid technique, but rather than learning to compensate for real-world model error, we learn a load model that allows us to further generalize the quasistatic model presented in [71] to various staged soft arm configurations.

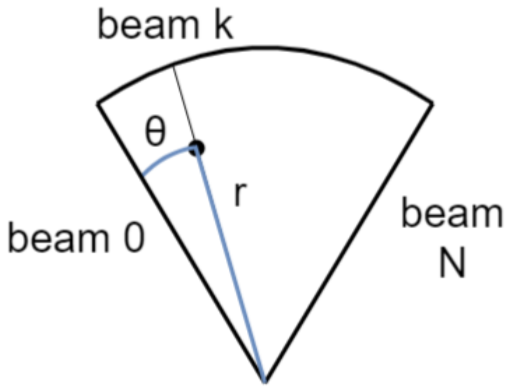
2.4 Sonar Based Underwater Mapping

2.4.1 Sonar Imaging

As seen in Fig. 2.1, multi-beam sonar creates images by sending out a series of acoustic beams and measuring the energy returned from a reflecting object. Typically the sonar return of a point in Euclidean space (X, Y, Z) is mapped to polar space (r, θ, ϕ). The range r to the object is obtained by measuring the time to return for the reflecting energy, the bearing angle θ to the reflector is characterized by the beam number k, and the pixel value (0-255) is mapped from the amount of energy received back. The elevation angle ϕ is lost in the imaging process.



(a) Computation of range and elevation angle



(b) The acquisition of bearing by emitting multiple beams

Figure 2.1: Mapping from Euclidean to polar coordinates.

2.4.2 Analyzing Sonar Imagery

Previous work in the area of sonar image analysis have used object shadows or strong image gradients to automatically identify features [38, 41]. While they report impressive results, we note that these features are not robust in our application. In the data we collected, oftentimes an object shadow was not present in the image. Additionally, in imaging moored objects, an object's shadow would not be present (due to the lack of reflection back from the seafloor around the object). During our deployments we also found that strong image gradients can lead to the appearance of features where there are none.

Recently, CNNs have been used to analyze sonar imagery. Kim et al. use CNNs to analyze sonar imagery and track the trajectory of another underwater vehicle [47]. While they are able to track this vehicle accurately, we compare to this method and show that the atrous architecture achieves a higher precision when classifying objects not in the training set. Williams and Dugelay address the problem of noise in sonar imagery by fusing together multiple views of the object of interest and using a deep network to classify images as either containing a man-made object or a naturally occurring rock [99]. We are able to complete our classification without the need for multiple views because our atrous network is able to account for noise in a single image.

There has also been work analyzing camera imagery captured onboard underwater vehicles. For example, Kaeli et al. identify a set of images that could be used to summarize the mission [43]. To identify images in this set they use Quantized Accumulated Histogram of Oriented Gradient (QuHOG) features, which rely on the gradients in the image. However, in sonar imagery the image gradients in noisy images can be deceptive features.

To address the noise and low-resolution of sonar imagery, we use the dilated filters in the atrous convolution architecture. In previous work we showed that the atrous architecture provides superior transfer learning capabilities when tested on images of objects not in the training set [17]. Details on this approach are discussed in Section 5.2.1.

2.4.3 Underwater Acoustic Structure From Motion

There has been much previous work in the area of using an imaging sonar for underwater 3D environment reconstruction. One popular approach in solving the underwater SLAM problem involves feature correspondence between pairs of sonar images. Hover et al. extract features from sonar images by clustering points in the image with large gradients [31]. A Normal Distribution Transform (NDT) is used for image registration, and the vehicle trajectory is optimized using a pose-graph. However, due to the unknown elevation angle of sonar image features, a planar assumption is used and the points are projected into the same plane as the vehicle. While this may work well for environments with large objects (e.g., sea floor, ship hull), this assumption is broken in more complex and unstructured environments.

The Acoustic Structure From Motion (ASFM) method proposed by Huang and Kaess provides two main advantages over other approaches [32]. First, ASFM relaxes the planar assumption for projecting sonar features into 3D. Second, their approach is capable of utilizing more than single pairs of sonar images to reconstruct 3D points. They formulate the reconstruction as a factor graph optimization problem. With the assumption of Gaussian noise, this can be formulated as a nonlinear least-squares problem. The factor graph is initialized by setting the sonar feature's unknown elevation angle to $0°$ and performing optimization via iterative linearization.

Huang and Kaess further extend their work to automatically perform feature association between sonar images [33]. Their proposed algorithm searches over a tree of potential feature correspondences, similar to Joint Compatibility Branch and Bound [65]. We leverage this work here to perform automatic feature correspondence and 3D reconstructions from hand annotated images. We contribute a complementary framework for analyzing incoming imagery to allow for the human to select the features in real-time.

Chapter 3: LIDAR Based Odometry Estimation

One challenging problem in robotics is accurate position estimation. Modern sensors are capable of providing high fidelity data; however, current CPU based processing methods are unable to take advantage of this data in real-time. For instance, the Velodyne HDL-32E can output nearly 700,000 laser points per second. An example scan generated with the HDL-32E can be seen in Fig. 3.1, illustrating the sensor's precision and high data output. Denoising autoencoders (DAEs) are a type of deep network originally used to learn feature representations robust to partial input corruption in image classification tasks [94]. We propose leveraging DAEs to capitalize on GPU processing speeds and extract salient features from projected sensor data to improve SE(3) motion estimation of a mobile robot using LIDARs such as these.

Referring back to Fig. 1.1, we can see that the data precision available in this domain suggests augmenting with optimization based methods. As such, we can take advantage of the many state-of-the-art algorithms for estimating odometry that use scan matching methods. While these methods can achieve accurate results, their iterative nature does not scale well with large amounts of data. Additionally, if data points are stored for use in a batch processing method, the memory footprint can quickly become intractable. These challenges pose an issue in applications requiring real-time operation or with strict hardware limits.

It is only recently that deep learning has begun to gain traction in robotics applica-

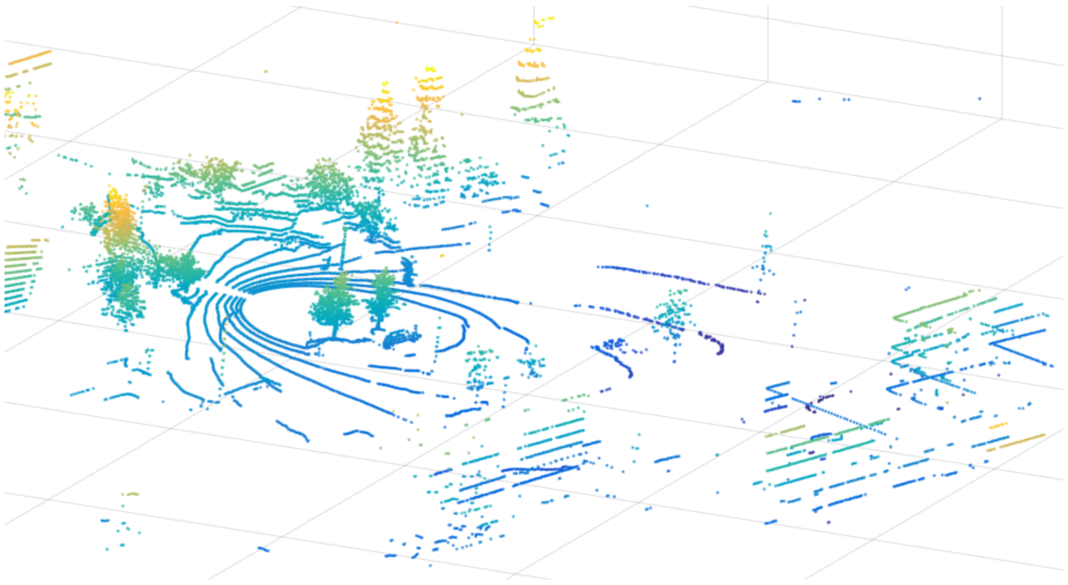


Figure 3.1: A single Velodyne HDL-32E scan from the NCLT dataset [11]. This particular scan contains 31,011 data points. Environmental features such as trees and buildings can be observed. The sparse nature of LIDAR at long range can be seen as well.

tions, notably in visual odometry [10, 70]. While these applications take advantage of the rich features present in RGB image data, LIDAR data presents unique challenges. One challenge is that real-world environments are complex and dynamic. That is, even if the sensor has not moved, a single beam may not measure the same point in physical space (e.g., pedestrians, moving vehicles). Additionally, the radial nature of the individual laser orientations can result in sparse data points, especially at long range. This is illustrated in Fig. 3.1, most notably visible in the vertical direction. Finally, the data throughput and range capabilities of modern LIDARs make it difficult to efficiently and accurately represent the data.

In this research, we show that through using top-down projections to efficiently represent LIDAR data, we are able to extract salient features from the data and more accurately estimate SE(3) motion for a mobile robot. In this projected image space, salient features of the LIDAR data are locally correlated. From Fig. 1.2, we can see that CNNs are ideal for this application. We show that our proposed approach is capable of improving the performance of standard scan matching techniques to be competitive with the state-of-the-art technique presented in [3]. Additionally, our proposed approach is capable of reducing the computation time required by state-of-the-art techniques with minimal loss in accuracy.

We train and validate our approach using real-world experimental data from the University of Michigan North Campus long-term vision and LIDAR dataset (NCLT Dataset) [11]. Benefits of this dataset include long-term, dynamic data in both indoor and outdoor environments as well as a robust post-processed SLAM solution for ground truth. We further test generalization (with no additional training) using LIDAR data from the KITTI dataset [23].

In summary, we present a novel pre-processing step for scan matching that:

- Removes outlier returns from raw point clouds, enabling faster, and depending on the technique used, more accurate scan registration.
- Takes advantage of convolutional layers to learn robust features from which outlier returns can be distinguished from small environmental features.
- Can be trained in an unsupervised manner, without requiring extensive hand labeling by a human expert.

The remainder of this chapter is organized as follows. We present a formulation of the problem in Section 3.1. Next, we describe our proposed approach in Section

3.2. Finally, we present our experiments and their results in Section 3.3. Versions of the proposed algorithm and results in this chapter have appeared in our prior workshop paper [69] and journal article [67].

3.1 Problem Formulation

In scan matching, the goal is to compute a transformation between two point clouds. In the standard point-to-point ICP algorithm presented by Besl and McKay [4], this is done by iteratively computing the correspondences between the point clouds. The transformation is estimated by minimizing the distance between corresponding points in the point clouds. For two point clouds, p and q, the transform from p to q, $\hat{T}_{pq}$, is given by:

$$\hat{T}_{pq} = \underset{T_{pq}}{\operatorname{argmin}} \sum_{i=1}^{N_p} \|T_{pq} p_i - q'_i\|^2 \tag{3.1}$$

where q' represents the set of corresponding points from q to p, p_i and q'_i represent the ith point in the respective point clouds, and N_p represents the number of points in cloud p.

The formulation presented by Besl and McKay was extended by Chen and Medioni [14] to match individual points to approximate surface planes. This is known as point-to-plane ICP. In point-to-plane ICP, rather than minimizing Euclidean distance, the distance is minimized along the surface normal. The optimization problem in Eq. 3.1 becomes:

$$\hat{T}_{pq} = \underset{T_{pq}}{\operatorname{argmin}} \sum_{i=1}^{N_p} \left\| (T_{pq} p_i - q_i') \cdot n_{q_i'} \right\|^2 \tag{3.2}$$

where $n_{q_i'}$ represents the estimated surface normal for individual point q_i'. The rest of the notation follows from Eq. 3.1.

There have been many other variants of ICP proposed. One such method is a generalized *plane-to-plane* ICP that models the surface planes for both p and q. This further increases the estimate quality by making the problem more robust to noise and/or outliers [82]. Other such methods employ *bundle adjustment* in which pairwise registrations are solved jointly [92]. However, this method is limited due to high computation and memory requirements.

In this work, we improve the accuracy and speed of ICP estimates by improving the quality, and reducing the size, of the input point clouds. That is, given an input point cloud p, our pre-processing step, $f(p)$, produces a new point cloud $\hat{p}$. In designing $f(p)$, the goal is to produce the smallest $\hat{p}$ possible such that $\hat{p}$ retains only the salient features in p.

3.2 Methodology

Our proposed approach updates the standard scan matching pipeline of directly aligning raw point clouds to include a DAE based pre-processing step. During this step, data is projected top-down using Eqs. 3.3-3.5 and fed into the DAE in image format. The DAE extracts robust features from the point cloud, with the output being used as a binary mask on the raw data projection. The masked data is then re-projected into 3D space by inverting Eqs. 3.3-3.5 and scan matching is performed in full 3D space. We note that throughout pre-processing, data is maintained in the sensor's local coordinate frame, allowing direct input to an ICP variant once re-projected. An overview of our approach can be seen in Fig. 3.2.

3.2.1 Data Representation

Ideally, feature extraction would occur directly in 3D space. For a 3D CNN, this can take the form of a voxel grid. This has been previously proposed for extracting features to perform terrain classification [64]. Others have used voxel grids to extract generic features from point clouds [22].

However, when discretizing large continuous spaces, dimensionality becomes an issue. Modern high fidelity LIDARs are capable of providing returns at ranges on the order of 100 m. At this range, discretizing the sensor's entire field-of-view (FOV) at centimeter resolution would require more than 1 billion voxels. Even if discretized to only meter resolution (far too coarse for accurate scan registration), over 1 million

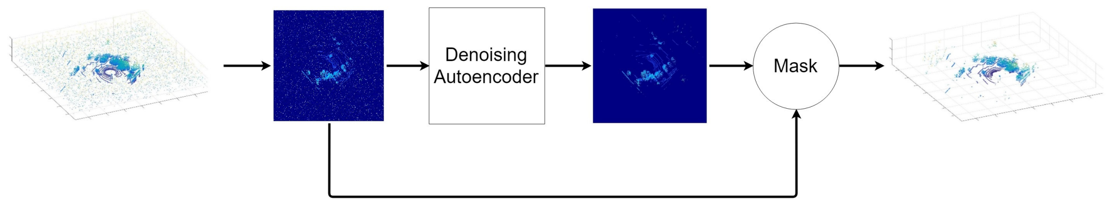


Figure 3.2: A block diagram representation of our DAE based pre-processing stage. A raw point cloud is projected top-down into image format and fed through the DAE. The DAE output is then used as a binary mask on the raw data projection. The masked data is then re-projected into 3D space.

voxels would still be required.

A more tractable approach is to utilize standard image classification techniques. To do this, we can modify our point cloud into a 2D projection of the environment; that is, a projected depth image. We do this by binning the raw LIDAR returns into a 2D pixel representation. Using this depth image and standard 2D convolutional layers, we can extract spatial features from the environment. This process is explained in more detail in the following sections.

3.2.2 Data Pre-Processing

We prepare our data for input into the deep network by projecting the raw LIDAR returns top-down. This projection technique is similar to the cylindrical projection that is used in [57]. With knowledge of the LIDAR's characteristics, we can use the following projection functions,

$$\mathit{row} = \left\lfloor y_{range} - \left(\frac{y}{\Delta y}\right) \right\rfloor \tag{3.3}$$

$$\mathit{col} = \left\lfloor \left(\frac{x}{\Delta x}\right) + x_{range} \right\rfloor \tag{3.4}$$

$$h = \frac{z}{z_{range}}, \tag{3.5}$$

where (x, y, z) represent a LIDAR return's 3D spatial coordinates and (col, row) represent the 2D pixel location of the projection. Δx and Δy represent the discretization along the x-axis and y-axis respectively. x_{range}, y_{range}, and z_{range} represent the range boundary to consider. Once the projected pixel location has been calculated, we fill its value with the point's normalized height, h. In this work, we set x_{range} and y_{range} to 75 m and Δx and Δy to 25 cm. We found this range captures the important environmental features while allowing for a more fine discretization at the pixel level.

In the case that two LIDAR returns are projected into the same pixel, we store the average of their heights. Pixels in which no returns are projected are filled with $h = 0$, so as to maintain sparsity in the image space. An example top-down projection can be seen in Fig. 3.3.

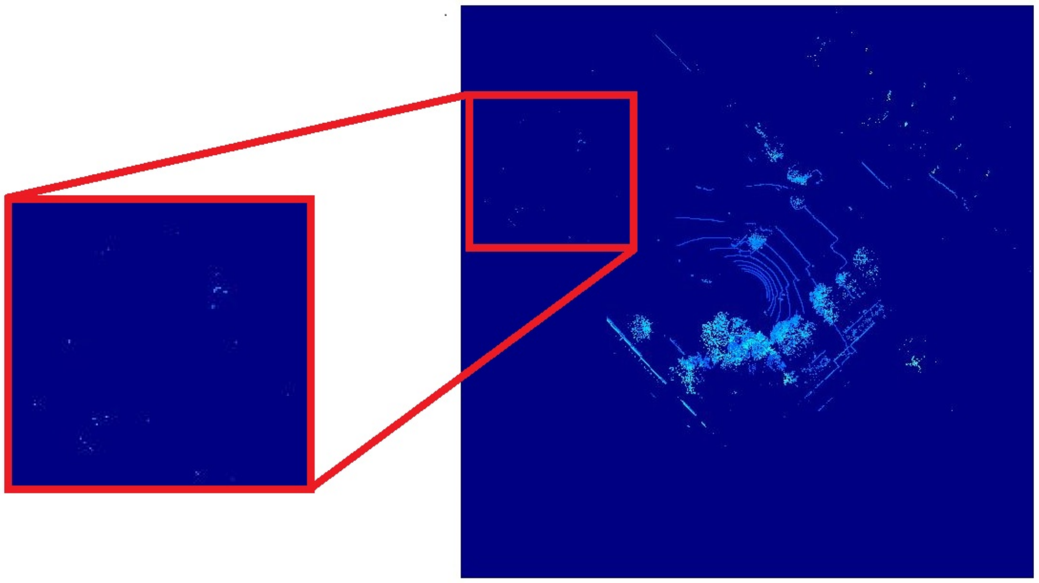


Figure 3.3: An example of the top-down projection used in this work. The environmental features of trees and buildings can still be seen. The zoomed area highlights outlier returns appearing as noise that can be attributed to both long range sparsity and the dynamic nature of the environment.

3.2.3 Network Architecture

DAEs attempt to learn robust representations of data by training the network to reconstruct the input data, x, from a corrupted version, $\tilde{x}$. This process is comprised of an encoder and a decoder. In the encoder stage, corrupted input data, $\tilde{x}$, is mapped to the hidden representation, y, as given by:

$$y = f(W\tilde{x} + b) \tag{3.6}$$

where the weight matrix, W, and offset vector, b, are learned parameters and $f(\cdot)$ represents a non-linear activation function. In the decoder stage, the hidden representation, y, is mapped back to a reconstructed version of the input data as $\hat{x}$:

$$\hat{x} = f(W'y + b') \tag{3.7}$$

where W' and b' again represent the learned weight matrix and offset vector, with $f(\cdot)$ representing a non-linear activation function. During training, the error between the uncorrupted input data, x, and the output reconstruction, $\hat{x}$, is used as the training signal:

$$L(x, \hat{x}). \tag{3.8}$$

In this work, we model our architecture on those used in RGB image classification (e.g., [83, 86]). As such, our network consists of stacked sets of convolution and pooling layers. With this structure, x represents the raw top-down scan projections while $\tilde{x}$ represents noise augmented scan projections. The weight and bias parameters, W and b, respectively, represent the learned convolutional filters. Note that for each convolutional layer, multiple filters are learned. Dropout and regularization parameters are further used to tune the network.

During training, input scans are augmented with noise while target scans remain unmodified. For each input scan, noise is added in the form of additional returns in the point cloud before projection. That is, additional points are added to the point cloud at uniformly random (x, y, z) locations such that the (x, y, z) location falls within the LIDAR FOV. We found that augmenting input scans with an additional 10%-20% noise provides a reasonable trade-off between noise removal and performance. In this work, we augment input scans with 15% additional noise (e.g., for a point cloud of size 10,000 we add 1,500 additional points).

The DAE takes as input a single 2D LIDAR scan projection. These inputs are then down-sampled by sets of convolution/pooling layer pairs. The last set of convolutional layers have no pooling layer pair. The loss function used on the network was the binary cross-entropy of the individual pixel values. The final network architecture used can be seen in Fig. 3.4, and the final layer parameters can be seen in Table 3.1. In the decoder section of the DAE, these parameters are simply reversed as the scan is reconstructed.

The size of the input 2D LIDAR scan projections are 600x600 pixels. In the first convolution/pooling layer pair, a convolution kernel of size 3x3, filter bank of size 32, and pooling size of 2 is used. The second convolution/pooling layer pair similarly uses a convolution kernel of size 3x3 and pooling size of 2, but instead uses a filter bank of size 64. The last pair of convolution layers again use a kernel of size 3x3 with a filter bank of size 64. The final output of the network is the reconstructed 2D scan projec-

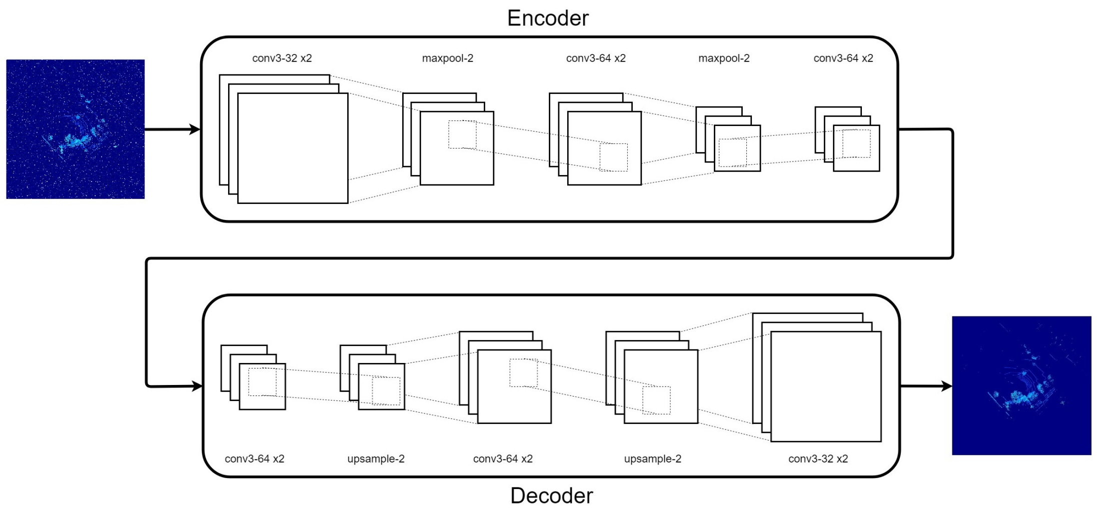


Figure 3.4: The autoencoder structure used in this work. An input image goes through a series of convolution and pooling layers before being reconstructed. The final reconstructed image is then used to re-project the LIDAR scan into full 3D space. Layer sizes and parameters can be seen in Table 3.1.

Table 3.1: DAE Encoder Parameters

Layer	**Dropout**	**Regularization Method**	**Regularization Value**
conv3-32	10%	L2	0.01
conv3-32	10%	L2	0.01
maxpool-2	∼	∼	∼
conv3-64	10%	L2	0.01
conv3-64	10%	L2	0.01
maxpool-2	∼	∼	∼
conv3-64	10%	L2	0.01
conv3-64	10%	L2	0.01

tion. The network uses dropout and L2 regularization on the convolutional layers with ReLU activations. During training, weights were randomly initialized and the Adam [48] optimizer was used. The network was trained for 250 epochs, or until converged (as determined via Keras early stopping criteria).

3.3 Experiments and Results

To demonstrate the capability of our approach, we perform two experiments on publicly available datasets. Experiment 1 demonstrates that our approach is capable of improving the registration provided by traditional scan matching techniques while reducing the computation time for all techniques. In Experiment 2, we show that the features learned by the DAE in Experiment 1 are robust and generalizable.

We compare the mean and max translational and rotational error components of the SE(3) registration as well as computation time. We decompose the rotation matrix into Euler angles. Statistical analysis is performed using one-way ANOVA tests. We note that all ICP implementations are CPU based. For methods including pre-processing, we include the pre-processing time in the timing metrics. Standard ICP methods are performed on raw point clouds. Variants including pre-processing are performed on processed point clouds.

We compare point-to-point ICP (*point-ICP*), point-to-plane ICP (*plane-ICP*), and a more state-of-the-art ICP implementation (*IRLS-ICP*) [3] with and without pre-processing. We test two pre-processing methods: ours and pre-processing from the Point Cloud Library (PCL) statistical outlier removal (*neighbors*=500, *threshold*=0.25).

3.3.1 Datasets

We leverage both the NCLT dataset [11] and KITTI dataset [23]. The NCLT dataset is aimed at facilitating research for long-term autonomous operation in changing environments. The KITTI dataset is comprised of data collected from a vehicle in a variety of environments (e.g., residential, highway). In this work, we use the *Residential* category.

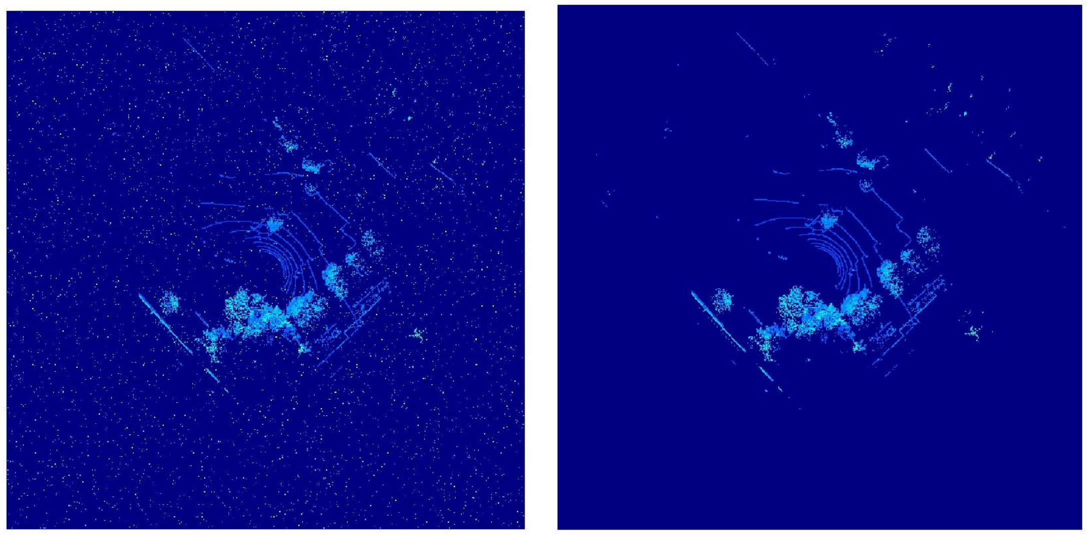
In the NCLT dataset, LIDAR data is provided by a Velodyne HDL-32E (32 lasers, 360° horizontal FOV, 41.3° vertical FOV, maximum return range of 100 m) while the ground-truth data is provided from a post-processed SLAM solution. In the KITTI dataset, LIDAR data is provided by a Velodyne HDL-64E (64 lasers, 360° horizontal FOV, 26.8° vertical FOV, maximum return range of 120 m) while the ground-truth data is provided GPS/IMU measurements.

3.3.2 Experiment 1: In-Domain Validation

This experiment demonstrates the capability of the DAE in extracting robust features from LIDAR point clouds. In order to train the DAE, a total of 3,000 scan projections were used. 750 scan projections from each of the *2012-01-08*, *2012-03-25*, *2012-06-15*, and *2012-10-28* NCLT data collection sessions were used. Each scan projection was randomly sampled from the first 80% of the data collection session. A total of 1,000 scan pairs were used to test the different methods. 250 scan pairs were randomly sampled from the remaining 20% of each of the data collection sessions.

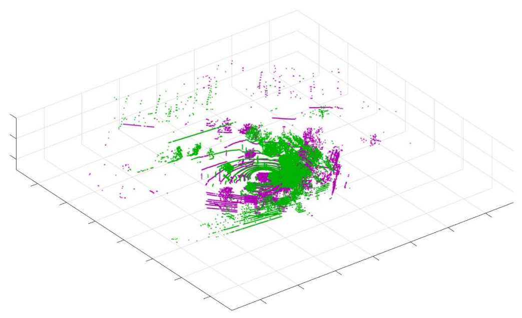
Fig. 3.5 shows a visualization of the DAE pre-processing step. Here, we can see that the DAE qualitatively removes noise from the input cloud. While most of the noise has been removed, we can see that small features, such as those in the upper right of the image, have been retained. In this case, these features appear to correspond to tall, thin objects (e.g., skinny tree trunks, light poles, etc).

Full results from the experiment can be seen in Table 3.2 with a visual comparison between the different methods presented in Fig. 3.6. From Table 3.2, we can see that the standard techniques perform the worst, most notably struggling with rotational accuracy. This rotational error can additionally be seen in Fig. 3.6b. In bootstrapping standard techniques with our DAE pre-processing step, *point-ICP* sees a moderate improvement ($p < 0.05$) in both translational and rotational errors while *plane-ICP* benefits the most

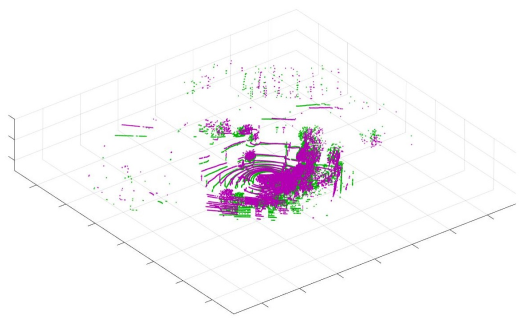


(a) (b)

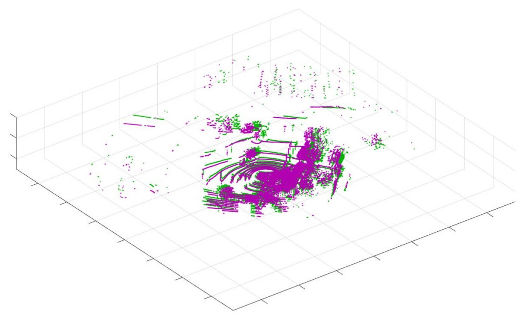
Figure 3.5: A visualization of the DAE performance. For visualization purposes, we add additional noise to the input scan. **(a)** The noise augmented scan projection. This projection is fed into the DAE for outlier removal. **(b)** The resulting masked scan projection. This projection is created by using the DAE output as a mask for the initial input scan, retaining only the salient features.



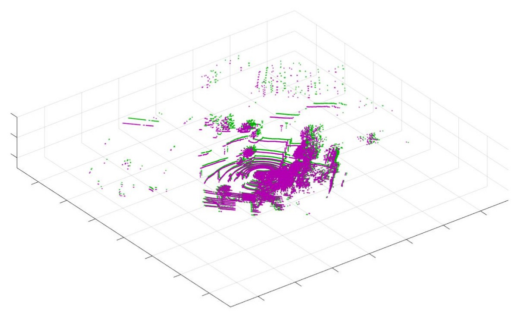
(a) Point clouds to align



(b) *Point-ICP*



(c) *DAE + plane-ICP*



(d) *IRLS-ICP* [3]

Figure 3.6: A comparison of the quality of the different techniques tested in this work. For each of the classes of techniques (standard, standard with pre-processing, and state-of-the-art), we present one visualization. For visualization purposes, the scans used here were separated by 2 sec. In (b), (c), and (d), the error between the estimated transform (green) and ground truth (purple) point cloud is shown. **(a)** The two raw point clouds to be aligned. Here, we wish to align the green cloud to the purple cloud. **(b)** The error using the standard *point-ICP* technique. **(c)** The error using the *DAE + plane-ICP* technique. **(d)** The error using the state-of-the-art *IRLS-ICP* [3] technique.

Table 3.2: Validation testing

	Mean Translational Error (m)	**Mean Rotational Error (deg)**	**Max Translational Error (m)**	**Max Rotational Error (deg)**	**Computation Time (sec)**
point-ICP	0.2391 ± 0.2061	2.1134 ± 1.5031	2.5370	7.2123	0.5956 ± 0.6514
plane-ICP	0.2452 ± 0.1692	2.1648 ± 1.5357	1.3395	7.4539	0.6150 ± 0.6686
PCL + point-ICP	0.2023 ± 0.1352	1.9275 ± 0.7089	2.2425	3.2102	0.5509 ± 0.5447
PCL + plane-ICP	0.1984 ± 0.1251	1.5501 ± 0.7998	1.2287	3.5406	0.6041 ± 0.6069
DAE + point-ICP (ours)	0.2112 ± 0.1960	1.2898 ± 0.7950	3.1644	4.0304	$\mathbf{0.2499 \pm 0.0402}$
DAE + plane-ICP (ours)	0.1764 ± 0.1474	1.3034 ± 0.8147	1.4299	4.8939	0.2874 ± 0.0368
IRLS-ICP [3]	0.1603 ± 0.0952	1.3402 ± 0.6303	0.6295	2.7987	51.2113 ± 36.9561
PCL + IRLS-ICP [3]	$\mathbf{0.1495 \pm 0.0726}$	1.2854 ± 0.5675	**0.4850**	**2.6144**	47.0437 ± 13.1895
DAE + IRLS-ICP [3] (ours)	0.1705 ± 0.1241	$\mathbf{1.2768 \pm 0.7936}$	0.8071	3.1315	8.1938 ± 1.8667

($p < 0.01$). We believe this is due to the improved quality of surface normal estimation. Also of note is that our pre-processing step produces more consistent results in the form of lower maximum errors.

However, we can also see that the IRLS-ICP algorithm performs the best with regards to accuracy. It is also the most robust in terms of maximum errors. This accuracy comes at a substantial cost, though, in computation time. Bootstrapping this technique with PCL pre-processing yields a statistically significant improvement in translational error ($p < 0.05$), but not rotational error. Bootstrapping IRLS-ICP with our DAE pre-processing yields no statistical difference in accuracy, but much lower computation time ($p < 0.01$).

We believe the speed increase can be attributed to two factors: First, the removal of outliers helps the registration process converge more quickly. Second, the reduction of total points in the clouds (due to both outlier removal and discretization) reduces the complexity of the optimization. Table 3.3 reports the mean point cloud size after the various methods of pre-processing. The processing time (per point cloud) of the PCL method and DAE are reported as well.

Table 3.3: Mean point cloud size reduction for the pre-processing methods used in this work. Associated processing times are reported in parentheses.

	Validation (pts)	**In-Domain (pts)**	**Cross-Domain (pts)**
None	50977	47084	127640
PCL	43594 (5.07 sec)	42578 (5.38 sec)	121560 (16.91 sec)
DAE	8805 (24.30 ms)	7847 (24.36 ms)	13183 (24.25 ms)

3.3.3 Experiment 2: In-Domain and Cross-Domain Generalization

This experiment demonstrates the generalization capabilities of the DAE. We test generalization in two cases: in-domain and cross-domain. For the in-domain test, we use the *2012-05-26* session from the NCLT dataset that was not seen by the DAE during training. 500 scan pairs were randomly sampled from the validation portion (final 20%) of the session for testing. For the cross-domain test, we use the *Residential* category of the KITTI dataset. 500 scan pairs were randomly sampled from the *2011-09-30* session.

In Table 3.4, we see that trends from Experiment 1 hold. This shows that the features learned by the DAE are not specific to the data collection sessions seen during training. Again, we see statistically significant improvement in translational error ($p < 0.05$), but not rotational error, for IRLS-ICP when PCL pre-processing is used. Our DAE pre-processing method sees no statistically significant change in accuracy, but requires much less computation time ($p < 0.01$).

Table 3.5 shows the results from the cross-domain test. We note that the difference in accuracy from previous tests can be attributed to the (generally) larger translational and lower rotational motion of the dataset (i.e., a car driving on roads). Additionally, the overall increase in computation time is due to the larger point clouds generated by

Table 3.4: In-domain generalization testing

	Mean Translational Error (m)	**Mean Rotational Error (deg)**	**Max Translational Error (m)**	**Max Rotational Error (deg)**	**Computation Time (sec)**
point-ICP	0.2189 ± 0.1213	1.6259 ± 1.2098	1.4823	6.1211	0.4397 ± 0.4933
plane-ICP	0.2448 ± 0.1096	1.6975 ± 1.2297	1.0713	6.2813	0.4413 ± 0.4771
PCL + point-ICP	0.1917 ± 0.1642	1.0866 ± 0.4907	1.2667	3.8108	0.4126 ± 0.3973
PCL + plane-ICP	0.1835 ± 0.1219	1.1529 ± 0.4724	1.3236	3.7216	0.4090 ± 0.4229
DAE + point-ICP (ours)	0.2084 ± 0.1293	1.0249 ± 0.6624	1.8599	3.7786	**0.1813 ± 0.0408**
DAE + plane-ICP (ours)	0.1711 ± 0.1478	1.0931 ± 0.7063	2.0108	4.9007	0.1932 ± 0.0327
IRLS-ICP [3]	0.1569 ± 0.0852	0.8502 ± 0.3289	0.4136	**2.1345**	41.5802 ± 28.8400
PCL + IRLS-ICP [3]	**0.1451 ± 0.0579**	**0.7525 ± 0.3521**	**0.3871**	2.3043	50.7275 ± 18.7314
DAE + IRLS-ICP [3] (ours)	0.1631 ± 0.0781	0.9310 ± 0.6683	0.4340	2.7533	4.4842 ± 0.8941

Table 3.5: Cross-domain generalization testing

	Mean Translational Error (m)	**Mean Rotational Error (deg)**	**Max Translational Error (m)**	**Max Rotational Error (deg)**	**Computation Time (sec)**
point-ICP	0.8963 ± 0.2051	2.3438 ± 0.4124	1.6020	2.3438	2.6760 ± 2.9494
plane-ICP	0.9787 ± 0.2493	1.6808 ± 0.5478	1.6808	5.0337	2.4964 ± 2.6810
PCL + point-ICP	0.7614 ± 0.2395	0.7176 ± 0.4733	1.4167	2.1647	2.5374 ± 2.5593
PCL + plane-ICP	0.7083 ± 0.1948	0.5241 ± 0.4092	1.3816	4.7302	2.3811 ± 2.4072
DAE + point-ICP (ours)	0.7859 ± 0.2243	0.8062 ± 0.3027	2.2034	2.4173	0.3387 ± 0.1146
DAE + plane-ICP (ours)	0.7364 ± 0.1566	0.5949 ± 0.5039	4.3084	6.0146	**0.3166 ± 0.0695**
IRLS-ICP [3]	0.3994 ± 0.0309	**0.2380 ± 0.1111**	0.8562	0.4692	132.1986 ± 12.1313
PCL + IRLS-ICP [3]	**0.3732 ± 0.0426**	0.2517 ± 0.1463	0.7838	**0.4251**	126.5397 ± 9.3756
DAE + IRLS-ICP [3] (ours)	0.4359 ± 0.1368	0.2862 ± 0.3692	**0.6774**	0.5948	15.9798 ± 4.1362

the Velodyne HDL-64E. The difference in point cloud sizes can be seen in Table 3.3. We note that as point cloud size increases, the DAE is able to maintain processing speed due to fixed input size.

In this experiment, the increased sensor movement between scans results in greater differences in accuracy between methods. Here, our DAE pre-processing method achieves a lower (but still competitive) accuracy than IRLS-ICP with statistical significance in both translational ($p < 0.01$) and rotational error ($p < 0.01$), as well as computation time ($p < 0.01$). Similar to previous experiments, PCL pre-processing outperforms IRLS-ICP in translational error ($p < 0.01$) but not rotational error or computation time.

Chapter 4: Hybrid Soft Robotic Control

Compared to traditional rigid robots, soft robots provide a unique set of challenges and benefits and are often used in low load, high flexibility tasks. Soft robots are seeing increased use across a variety of domains, including grasping [104, 62], prosthetic devices [84, 74], and biologically inspired locomotion [93, 6, 58, 8]. Currently, much of the research in the field tends to focus on improving the methods for building these types of robots (e.g., design or materials). Despite this, when compared to the previous chapter, the data obtained from these types of robots is less precise due to real-world disturbances (e.g., friction, manufacturing imperfections). As suggested by our framework in Fig. 1.1, augmenting with modeling based approaches may be more appropriate here. With this in mind, we can take advantage of recent improvements in generalized soft robotic modeling (e.g., [71]).

Soft bending arms are most commonly constructed with three or more longitudinal actuators in parallel, where bending is controlled by lengthening or shortening actuators. Biological systems, such as octopuses, use a similar arm architecture, with the arm tapering towards the tip [46]. Current state-of-the-art dynamic models for soft bending arms require experimental fits to determine their physical parameters. While these are suitable for controlling individual arms, varying the arm configuration requires a new set of physical parameters to be determined. A more generalized model is desirable to take advantage of the fact that arm configurations comprised of multiple link segments

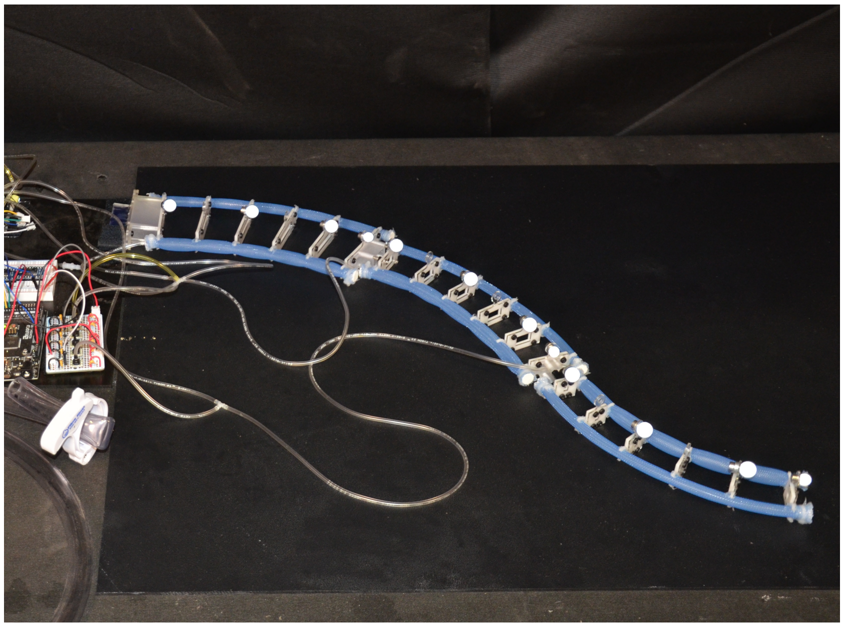


Figure 4.1: An example arm pose achieved by the 80-50-35mm staged arm configuration used in this work.

can be reconfigured to suit different scenarios. The model proposed in [71] combines conservation of energy with static equilibrium to provide a generalizable formulation; however, this formulation assumes an unloaded arm.

In order to work with the types of staged arm architectures previously mentioned, we need to be able to account for the system load introduced when connecting multiple longitudinal actuators in sequence. In this work, we propose an approach that augments the generalizable model in [71] with a data-driven, learned load model. With this approach, no additional learning or parameter fitting is required when switching between arm configurations. Another benefit is that the generalizable model provides a lower bound for our input control pressures that would otherwise not be available when learning the entire control model. In our approach, we first determine the staged arm configuration in use via a single-shot calibration routine that sweeps through the arm's entire range of motion in an OptiTrack system. From Fig. 1.2, we can see that an LSTM is ideal here due to the temporal nature of the motion data. After calibration, only simple pressure based on-off control is required to control the arm. Since this only requires a simple mapping function, Fig. 1.2 suggests a fully connected network. We validate the proposed approach on multiple staged soft arms showing that it is able to accurately identify the arm configuration in use. Additionally, we show that the learned load model is able to compensate for the system load across various staged arm configurations.

In summary, we present a novel control architecture for reconfigurable staged soft arms that:

- Is capable of compensating for the system load introduced when connecting multiple longitudinal soft actuators in sequence.
- Takes advantage of the temporal nature of the arm motion to accurately identify the arm configuration currently in use.
- Can switch between different arm configurations on the fly, requiring no additional learning or parameter fitting when doing so.

The remainder of this chapter is organized as follows. We describe the problem formulation in Section 4.1. Next, we discuss our proposed approach in Section 4.2. Finally, we present our experiments and their results in Section 4.3. The proposed approach and results presented in this chapter are under review in our journal paper [68].

4.1 Problem Formulation

The quasistatic model proposed in [71] is capable of providing a mapping between control input and soft arm states for individual unloaded link segments as given by

$$u = P_{qs}(\kappa, t), \tag{4.1}$$

where P_{qs} represents the quasistatic model and u, κ, and t represent the input control pressure, soft arm state, and soft arm width, respectively. In our proposed approach, we learn a model for the system load introduced when combining these individual link segments into a staged arm configuration. Following the previous notation, this extends our mapping function to

$$u_t = P_{qs}(\kappa, t) + P_{load_t}(s, \kappa), \tag{4.2}$$

where t represents the width of a given link segment in the staged arm and u_t, P_{load_t}, and s represent the input control pressure, learned load model, and load state for the link segment of width t, respectively. Our approach uses a fully connected deep network to learn the load model and an LSTM to identify the current load state. Because we have well defined, distinct arm configurations, we choose to input the load state classification (rather than a latent representation) into the fully connected network. This allows a more direct comparison between our fully connected network and curve fitting approaches (see Section 4.3) which also use load state as an input. A block diagram overview of our system can be seen in Fig. 4.2.

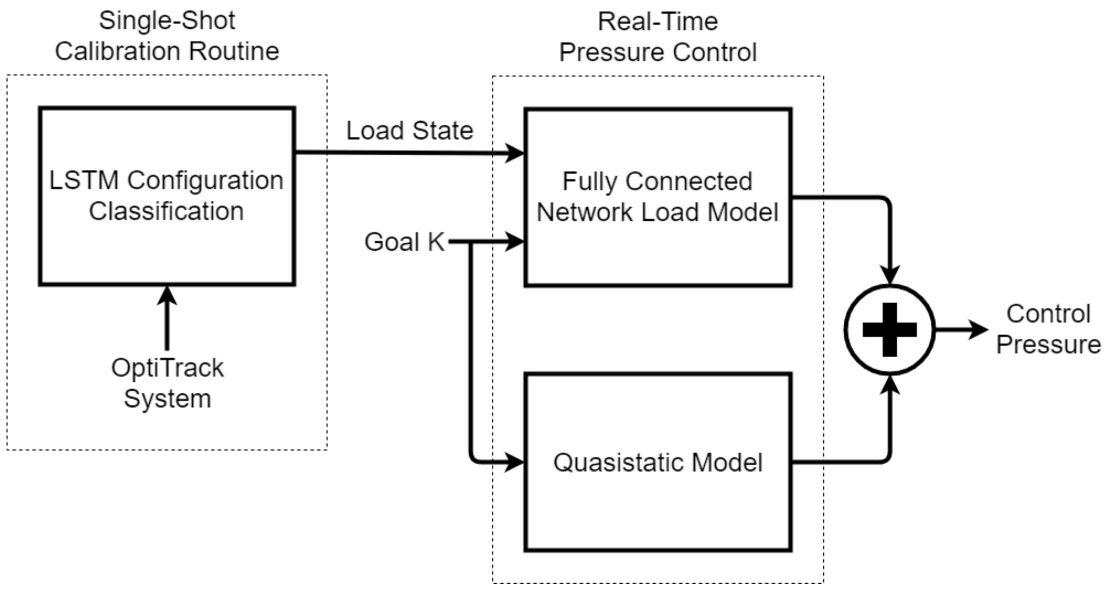


Figure 4.2: A block diagram of the architecture proposed in this work. The staged arm configuration is only determined once during the single-shot calibration routine. The predicted load state is then used for pressure control during operation. Only the calibration routine requires use of the OptiTrack system.

4.2 Methodology

In this section, we discuss the two main components of our approach. In Section 4.2.1, we discuss the single-shot calibration routine used to classify the load state of the arm. In Section 4.2.2, we detail how we use the quasistatic model and learned load model to control the staged arm.

4.2.1 Load State Estimation

Since we are using tapered staged arm configurations, every arm configuration produces a unique load state for each link segment in the arm. This allows us to identify every individual segment load state by simply determining which staged arm configuration is currently being used. Additionally, this enables us to identify the entire staged arm configuration by only looking at the base link of the arm.

We classify which configuration the arm is in by looking at how the base link control input, u_{base}, and state, κ_{base}, change with respect to time. In the absence of external disturbances, κ_{base} changes smoothly as u_{base} is increased. The result of this is that the pair $(u_{base}, \kappa_{base})_i$ is highly predictive of the pair $(u_{base}, \kappa_{base})_{i+1}$. As such, we use an LSTM based network for classification. To do this, we perform a single-shot calibration routine using an OptiTrack system. This calibration routine is comprised of a single sweep of the arm through its entire valid pressure range. During this sweep, $(u_{base}, \kappa_{base})$ pairs are obtained from the OptiTrack and actuators, respectively. These pairs of data are passed into the LSTM after which a configuration prediction is produced by selecting the class with highest probability.

Because an entire arm sweep is used as classification input, it is difficult to acquire a sufficient amount of training data by hand. To this end, we use a data augmentation technique to lower the amount of data that must be collected. We generate training data sequences by stitching together segments of recorded sweeps to form training sweeps. The process by which we do this can be seen in Algorithm 1. Because we directly stitch these segments together without smoothing, this additionally allows us to provide the network with training sequences that approximate real-world disturbances (e.g., friction, manufacturing imperfections) allowing our classifier to be more robust.

Algorithm 1: Training Data Generation

```
// recorded:  recorded data sequences
// N: training sequences to generate
// training:  generated data sequences
Input: $recorded$, $N$
Output: $training$
// set of all possible segment lengths
$seg\_len$ = {20% 30% 40%}
for $i$ = 1 to $N$ do
    $start\_idx$ = 0%                    // begin at start of sequence
    while $start\_idx < 100\%$ do
        $len \overset{R}{\leftarrow} seg\_len$                   // randomly select
        $seq \overset{R}{\leftarrow} recorded$                   // randomly select
        $end\_idx$ = $start\_idx$ + $len$
        if $end\_idx \geq 100\%$ then
            $end\_idx$ = 100%                // cut segment short
        end
        // get the segment, append to generated sequence,
           and update the index
        $segment$ = $seq[start\_idx:end\_idx]$
        $training_i$.append($segment$)
        $start\_idx$ = $end\_idx$
    end
end
```

4.2.2 Control Input Mapping

As previously mentioned, we decompose the problem of mapping soft arm states to control inputs into a quasistatic model mapping and load mapping: $u_t = P_{qs}(\kappa, t) + P_{load_t}(s, \kappa)$. This approach has several benefits. Adding a load to the system will always increase the control pressure required to achieve a given link curvature state. Because of this, the quasistatic model provides us with a lower bound for the control input. In only learning a portion of the overall state to control input mapping, we minimize the potential impact of incorrect predictions by the network. If we were to learn the entire mapping, we have no expectations as to the worst case arm behavior from an incorrect network prediction. Instead, with our method, we guarantee that at worst we follow the quasistatic model, but more likely than not move in some direction towards the desired behavior. Because we know the upper bound pressure ratings on our actuators, we can always guarantee a control input in the safe operating range.

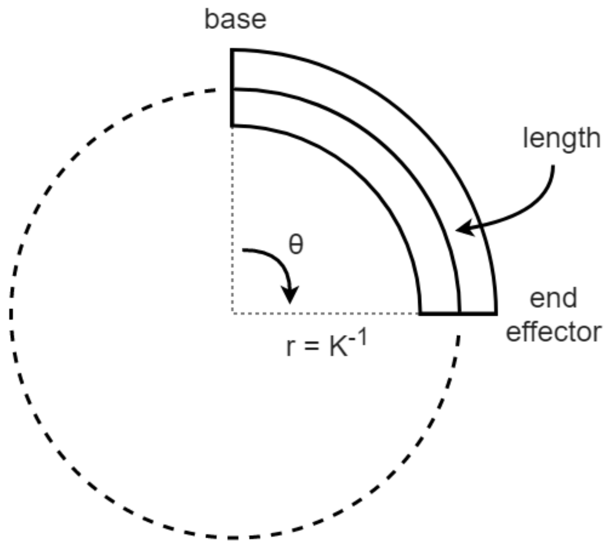


Figure 4.3: An illustration of the constant curvature assumption used to model the arm segments.

4.2.2.1 Quasistatic Model

The quasistatic model developed previously [71] was used in this work. The model assumes constant curvature (Fig. 4.3), an always planar cross section and no loads applied to each arm segment. The arms studied have three segments, each of which has two actuators. Only one actuator in each segment is pressurized at a time. We calculate curvature by combining moment equilibrium and conservation of energy with a kinematic constraint. The pressurized actuator generates a force F_P that is dependent on the known pressure P and unknown strain ε_P, while the passive actuator force F_E depends only on that actuator's strain ε_E . The moment balance for an arm of width t, summed about the unknown location of the neutral axis, h_P is

$$\Sigma M = -F_P(\varepsilon_P, P)(-h_P) + (-F_E(\varepsilon_E))(t - h_P) = 0. \tag{4.3}$$

Each actuator has a stored strain energy U_P and U_E, and the pressurized actuator produces work W_P. The energy balance for each segment is

$$\Sigma E = W_P(P, \varepsilon_P) - U_P(\varepsilon_P) - U_E(\varepsilon_E) = 0. \tag{4.4}$$

The system of equations is closed through a kinematic relationship, which relates the pressurized and passive actuators strains geometrically:

$$\varepsilon_P = -h_P\kappa \tag{4.5}$$

$$\varepsilon_E = (t - h_P)\kappa. \tag{4.6}$$

The actuator force, work and strain energy are calculated from experimental force characterizations [71].

4.2.2.2 Load Model

In order to learn the mapping for the load model we utilize a fully connected deep network. That is,

$$P_{load_t}(x) = f(Wx + b), \tag{4.7}$$

where the weight matrix, W, and offset vector, b, are learned parameters; $f(\cdot)$ represents a non-linear activation function; and the input x is given by (s, κ). We train this network on real-world data collected from the arm configurations used in this work. An individual deep network is trained for each of the link segment widths used in our arm configurations. We note that this scales linearly with the number of link segment widths available. Increasing the length of the overall arm (as measured in number of link segments) does not require additional networks to be trained, but rather increases the number of load states to model.

Training data is generated by controlling the individual link segments to various curvatures and recording the corresponding actuator pressure state. The quasistatic model value is then subtracted from the recorded pressure to obtain the load model value to learn. In order to efficiently generate training data, we perform arm sweeps that cover the entire valid pressure range of the actuators. This is beneficial for two reasons. First, as the arm sweeps through the pressure range, we generate many pairs of training data without needing to wait on a controller settling time as it reaches a specified curvature. Second, every sweep of the arm performs slightly differently due to an accumulator tube and real-world noise (e.g., friction and manufacturing imperfections). This slight performance variation naturally obtains data samples at different curvature values throughout the valid pressure range. The entire training dataset is generated by combining all recorded data points into a single distribution which can be modeled as

$$(\mathbf{u}, \boldsymbol{\kappa}, \mathbf{s}) = P_{load_t} + N(\mu, \sigma^2), \tag{4.8}$$

where ($\mathbf{u}$, $\boldsymbol{\kappa}$, $\mathbf{s}$) is the set of all data samples, P_{load_t} is the true load model we wish to learn, and $N(\mu, \sigma^2)$ approximates the real-world noise observed.

4.3 Experiments and Results

We demonstrate the capability of our approach through three experiments. In Experiment 1 we evaluate the classification capabilities of various network types and show that using an LSTM is an intuitive approach capable of high classification accuracy. Experiment 2 demonstrates that a deep network is capable of accurately modeling the system load while providing the scalability necessary for larger staged arms. Finally, Experiment 3 illustrates the overall increase in end-effector accuracy achieved by our approach as compared to quasistatic model only control.

In all three experiments, we use four different link segment widths: 80 mm, 65 mm, 50 mm, and 35 mm. Staging these four widths in tapered configurations provides us with three unique arm configurations: 80-65-50mm, 80-65-35mm, and 80-50-35mm. Each arm segment is made of two actuators joined by six evenly spaced radial support plates. Actuators were made using EcoFlex 00-30, had a wall thickness of 1.5 mm, internal diameter of 6 mm, length of 240 mm, and were reinforced with polyester expandable sheathing (McMaster #9284K2). For full manufacturing details, we refer the reader to [71].

In Experiment 1, we compare two classification methods: a fully connected deep network and an LSTM. We use precision and recall as our metrics for success. In general, precision and recall are calculated as

$$Precision = \frac{T_p}{T_p + F_p} \tag{4.9}$$

$$Recall = \frac{T_p}{T_p + F_n}, \tag{4.10}$$

respectively, where T_p represents true positives, F_p represents false positives, and F_n represents false negatives. For multi-class problems, the precision of the i^{th} class is calculated as

$$Precision_i = \frac{C_{ii}}{\sum_j C_{ji}}, \tag{4.11}$$

where C denotes the confusion matrix with i and j representing the row and column dimensions, respectively. Using the same notation, recall is given by

$$Recall_i = \frac{C_{ii}}{\sum_j C_{ij}}. \tag{4.12}$$

In Experiment 2, we measure accuracy in curvature space using the mean and maximum error. For each segment width, all load cases are averaged into the reported metric. We compare several methods:

- **Curvature Control (baseline)** uses the OptiTrack system for state feedback, directly controlling the arm to the desired curvature. This represents the maximum accuracy of our controller.
- **Individual Curve Fit Control** uses individually fit curves (6th order polynomial) to model each load state of each segment width. This approach scales combinatorically with both the number of link widths and arm configurations considered.
- **Single Surface Fit Control** uses a single surface fit (6th order polynomial) to model all load states for every segment width. This approach scales linearly with the number of link widths.
- **Deep Network Control** uses a single trained deep network to model all load states for each segment width. This also scales linearly with the number of link widths.

In Experiment 3, mean and maximum errors of the end-effector are reported in euclidean space to provide a more intuitive understanding of the quality of control. We compare three methods:

- **Curvature Control (baseline)** represents the maximum accuracy of our controller.
- **Quasistatic Model Only Control** uses only the model proposed in [71] to control the arms and does not account for load.

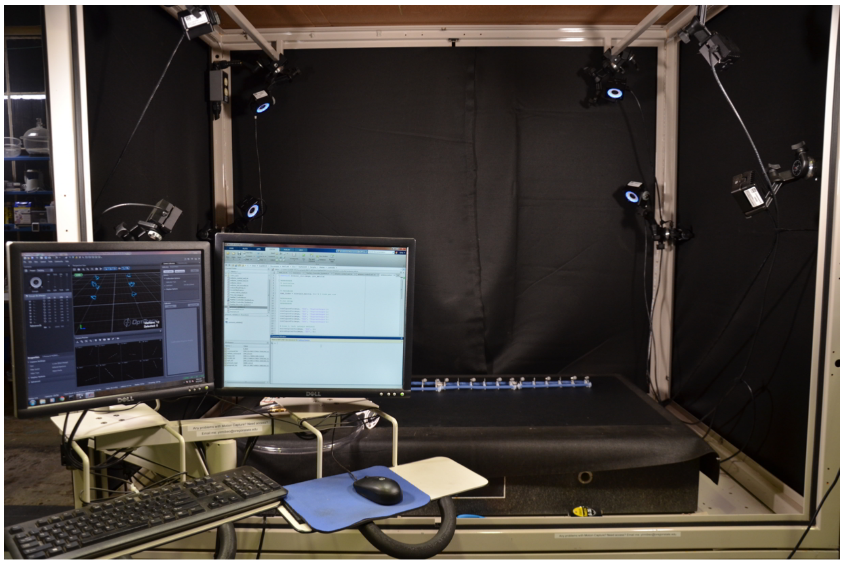
- **Quasistatic Plus Load Model Control** uses the proposed approach to control the arm, accounting for load using a fully connected deep network.

4.3.1 Hardware Setup

For this work, full arm control was achieved by controlling individual link segments with on-off feedback control. That is, for each individual segment, a desired goal state is specified. The individual segment actuators are then inflated or deflated based on their current state. Using a custom control board, our controller is capable of both pressure and curvature control.

The control board is comprised of a diaphragm pump, solenoid valves, pressure sensors, and an Arduino. The single diaphragm pump provides airflow to the entire system while the solenoid valves control whether or not individual link segment actuators inflate, deflate, or maintain pressure. The Arduino is used to control the diaphragm pump speed as well as the solenoid states. MATLAB is used as the interface to communicate with both the control board and the OptiTrack system. System feedback is available in two forms: pressure and curvature. Pressure feedback is obtained via onboard Honeywell Basic ABP Series sensors. Curvature feedback is obtained from the positions of sets of retroreflective markers as given by the OptiTrack system. The hardware setup can be seen in Fig. 4.4.

Our proposed approach operates using only pressure control. This allows for a greater range of applications by not requiring our arm to operate in the OptiTrack staging area. In Experiment 2 and 3, we compare our proposed approach against curvature


(a)

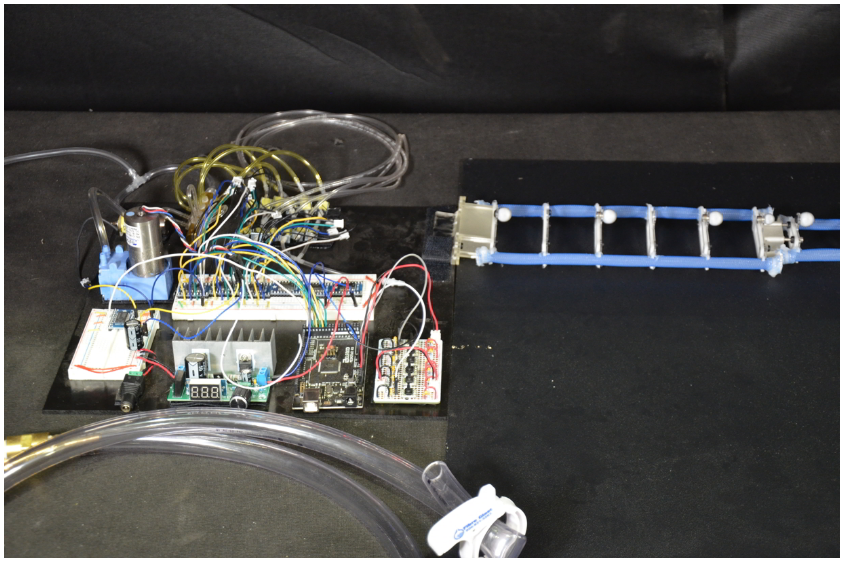

(b)

Figure 4.4: The hardware setup used in this work. **(a)** The full OptiTrack staging area. The left monitor displays the Motive software while the right monitor displays the MATLAB control interface. **(b)** A close up of the control board, accumulator tube, and arm mount.

control as a means of comparing against the maximum accuracy of our controller. In this method, rather than generating the control input pressure to achieve a desired curvature, we bypass the prediction step and control to the desired curvature directly.

4.3.2 Experiment 1: Configuration Classification Accuracy

This experiment illustrates the classification capability of our method. We compare our proposed LSTM architecture against a standard, fully connected deep architecture. The deep architecture is chosen for comparison to represent a brute force approach to the learning problem that does not take advantage of the temporal nature of our data. The specific architecture parameters can be seen in Tables 4.1 and 4.2. The layer naming convention denotes both the type of layer and number of units (i.e., *lstm30* represents an LSTM layer with 30 units). We train and validate on 600 and 150 sweeps per arm configuration, respectively (total: 1800 and 450). The networks were trained for up to 500 epochs (with early stopping) using the Adam optimizer [48].

Full results from the experiment can be seen in Table 4.3. We can see that both network architectures were able to achieve high classification accuracy, with the LSTM

Table 4.1: LSTM
Classifier Architecture

Layer	**Activation**	**Dropout**
lstm30	ReLU	$\sim$
lstm30	ReLU	$\sim$
dense3	Softmax	$\sim$

Table 4.2: Fully Connected Classifier Architecture

Layer	**Activation**	**Dropout**
dense512	ReLU	20%
dense256	ReLU	20%
dense64	ReLU	20%
dense3	Softmax	$\sim$

Table 4.3: Classification Accuracy Results. Both the mean and individual class metrics are reported. The individual classes are specified by the arm configuration details (mm).

	Precision				**Recall**			
	Mean	**80-65-50**	**80-65-35**	**80-50-35**	**Mean**	**80-65-50**	**80-65-35**	**80-50-35**
Fully Connected	94.85%	100.0%	88.89%	95.65%	94.67%	100.0%	96.0%	88.0%
LSTM (proposed)	97.33%	100.0%	96.0%	96.0%	97.33%	100.0%	96.0%	96.0%

slightly outperforming the fully connected deep network. In both cases, the 80-65-50mm configuration was easiest to classify. This is likely because the 35 mm segment has the largest range of motion, and it is easy to detect the configuration without that segment. Classifying between the 80-65-35mm and 80-50-35mm configurations proved more difficult as the difference in motion between the 65 mm and 50 mm segments is more slight (especially when loaded with the 35 mm segment). Additionally, the LSTM network required fewer trainable parameters (11,373 vs. 250,883) and converged faster (287 vs. 326 epochs) to achieve the reported results.

4.3.3 Experiment 2: Load Model Accuracy

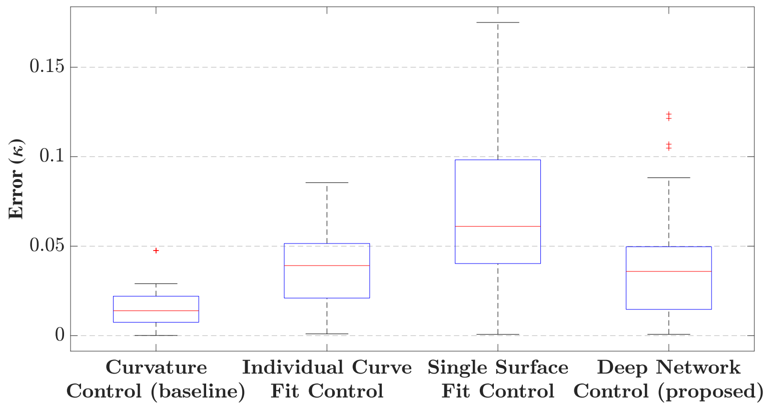
In this experiment, we show the ability of our method to learn the model required to account for the system load introduced in a staged arm. As previously noted, each staged arm configuration maps to a unique load state for the individual links that comprise it. For this experiment, both the deep network and surface fit are given the ground truth arm configuration state so as to directly measure the quality of the load model learned. The deep network architecture and parameters can be seen in Table 4.4. We train and validate on 1000 and 200 data points per load case (total: 3000 and 600 for the 80 mm segment, 2000 and 400 for the 65 mm segment, 1000 and 200 for the 50 mm segment). We train the network for up to 200 epochs using the Adam optimizer [48].

We control each loaded segment (i.e., all segments except for the 35 mm segment) to 25 distinct curvatures spaced along the segment's range of motion. We note that this range of motion changes for each segment width and load condition, and thus different curvatures are chosen for each condition.

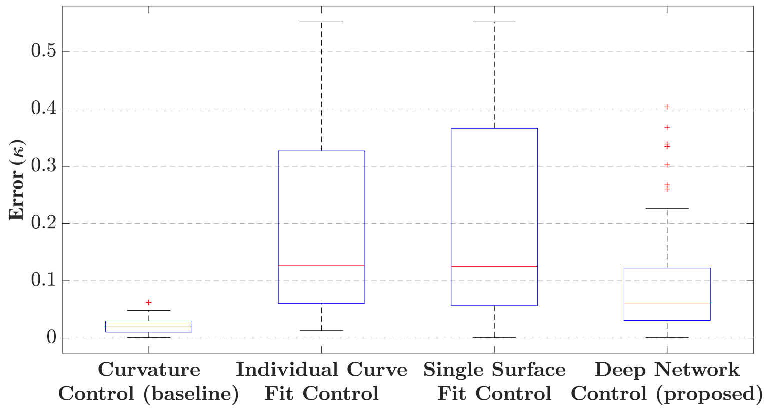
The results of this experiment can be seen in Table 4.5 and Fig. 4.5. From the table, we can see that direct curvature control provides a very accurate baseline. The individual curve fits and our proposed method perform accurately (and similarly, except for the 65 mm segment). While the performance is similar, as previously mentioned, the individual curve fit method scales combinatorically with the number of link widths and arm configurations used. Despite the single surface fit performing the worst in all cases,

Table 4.4: Fully Connected
Load Model Architecture

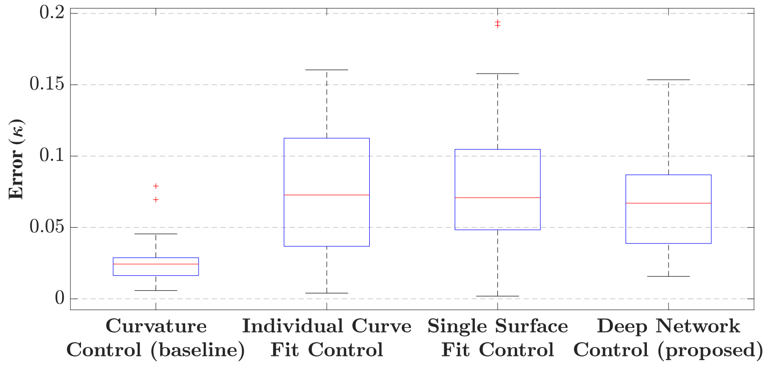
Layer	**Activation**	**Dropout**
dense128	ReLU	10%
dense32	ReLU	10%
dense1	Linear	$\sim$



(a) 80 mm link



(b) 65 mm link



(c) 50 mm link

Figure 4.5: A box and whisker plot comparison of the results for Experiment 2. Lower is better. For full results, see Table 4.5.

Table 4.5: Individual Link Control Accuracy Results. Details regarding each method listed can be found in Section 4.3.

	80 mm Segment		**65 mm Segment**		**50 mm Segment**	
	Mean Error (κ)	**Max Error (κ)**	**Mean Error (κ)**	**Max Error (κ)**	**Mean Error (κ)**	**Max Error (κ)**
Curvature Control (baseline)	0.015 ± 0.011	0.047	0.022 ± 0.016	0.062	0.026 ± 0.017	0.079
Individual Curve Fit Control	**0.037 ± 0.021**	**0.086**	0.189 ± 0.152	0.552	0.076 ± 0.047	0.160
Single Surface Fit Control	0.071 ± 0.041	0.175	0.204 ± 0.177	0.551	0.082 ± 0.052	0.194
Deep Network Control (proposed)	**0.037 ± 0.028**	0.123	**0.103 ± 0.108**	**0.403**	**0.072 ± 0.041**	**0.154**

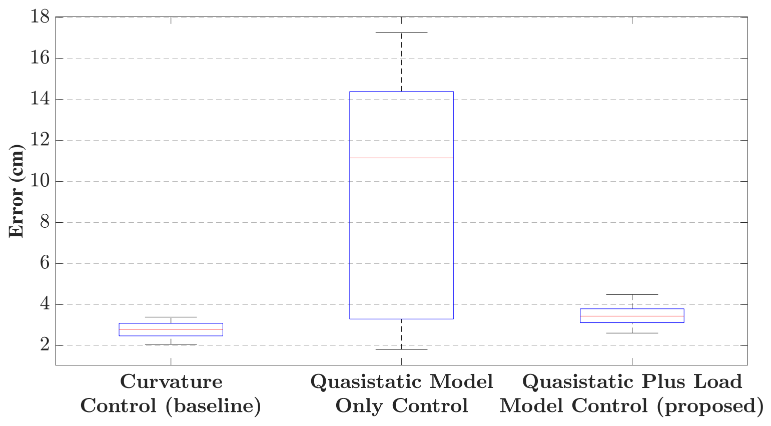
the gap between it and other methods decreased as the number of load cases modeled decreased (i.e., segment width decreased). This makes sense since for fewer load cases, the modeling problem is more straightforward.

We note that the 65 mm arm segment was the hardest to control, experiencing inconsistent performance in the low pressure region. Notably, this affected the performance of the individual curve fits more than our proposed method. The performance difference can be seen in both the mean and maximum error rates. This is likely due to the fact that the deep network can model an arbitrarily more complex function than the curve fitting toolbox.

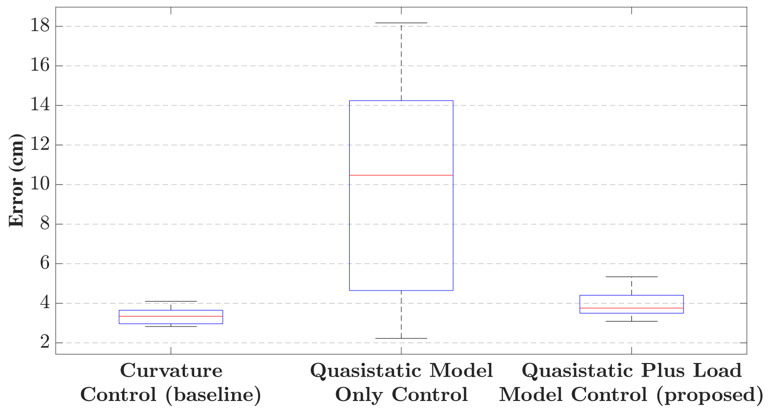
4.3.4 Experiment 3: End-Effector Accuracy

This experiment demonstrates the capability of the overall system proposed. In this experiment, we first perform the single-shot calibration routine. This is done by placing the staged arm in the OptiTrack staging area and performing a single sweep of the base link segment's full pressure range (0-12 psi). The recorded control input and curvature state data, $(u_{base}, \kappa_{base})$, is passed through the LSTM classifier. The load state prediction for each link in the staged arm, $\hat{s}$, is given by the maximum class probability. This load state prediction is then used for the remainder of the experiment, even if misclassified, so as to fully capture the accuracy of the proposed system. The load state and desired curvature state are then used to generate control inputs, $u_t = P_{qs}(\kappa_{goal}, t) + P_{load_t}(\hat{s}, \kappa_{goal})$ for each link in the arm configuration. For this experiment, we control the end-effector to 25 distinct, randomly generated arm states within the reachable workspace for all arm configurations.

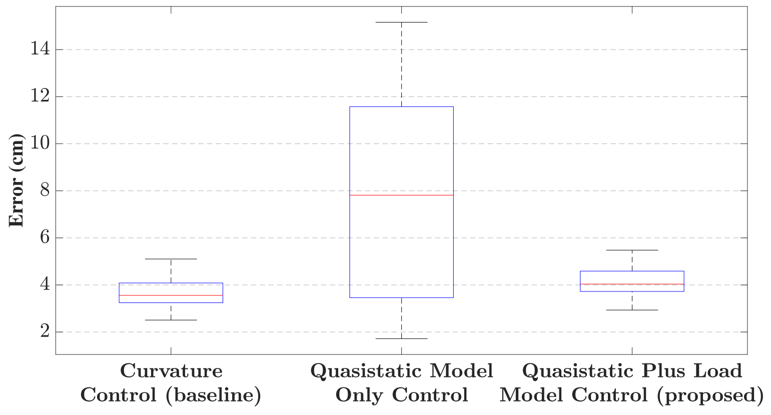
The results from this experiment can be seen in Table 4.6 and Fig. 4.6. The table shows, as expected, that the curvature control baseline is the most accurate. Even though the mean error appears to be somewhat high, we note that this error largely comes from the shortening of the base curve of the link segments as they bend. While the control is highly accurate in curvature space (as shown in Experiment 2), as the link bends and shortens, error is introduced along the segment arc. This is further evidenced by the low standard deviation in the results, demonstrating the consistency of both the baseline and our approach. The link shortening can be seen in Fig. 4.7 and is further discussed in [71].



(a) 80-65-50mm arm



(b) 80-65-35mm arm



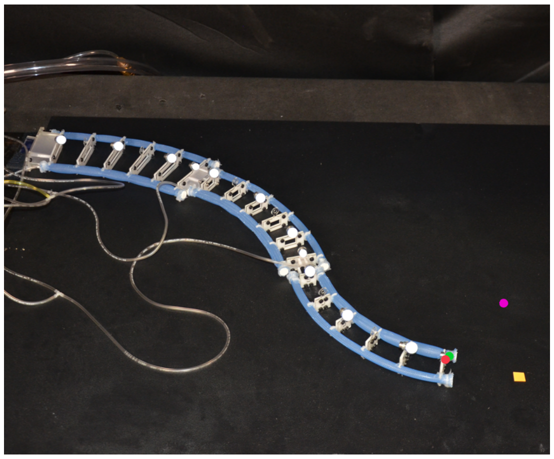
(c) 80-50-35mm arm

Figure 4.6: A box and whisker plot comparison of the results for Experiment 3. Lower is better. For full results, see Table 4.6

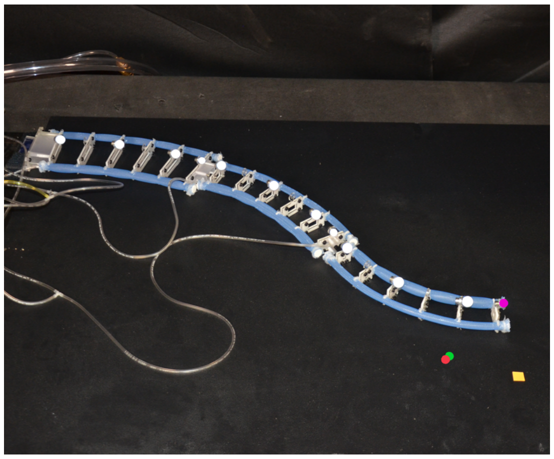
Table 4.6: Full Arm Control Accuracy Results. Details regarding each method listed can be found in Section 4.3.

	Arm 80-65-50		**Arm 80-65-35**		**Arm 80-50-35**	
	Mean Error (cm)	**Max Error (cm)**	**Mean Error (cm)**	**Max Error (cm)**	**Mean Error (cm)**	**Max Error (cm)**
Curvature Control (baseline)	2.791 ± 0.380	3.386	3.342 ± 0.402	4.099	3.672 ± 0.626	5.103
Quasistatic Model Only Control	9.215 ± 5.535	17.267	9.572 ± 5.484	18.171	8.110 ± 4.457	15.162
Quasistatic Plus Load Model Control (proposed)	**3.468 ± 0.512**	**4.490**	**3.9282 ± 0.602**	**5.339**	**4.125 ± 0.698**	**5.479**

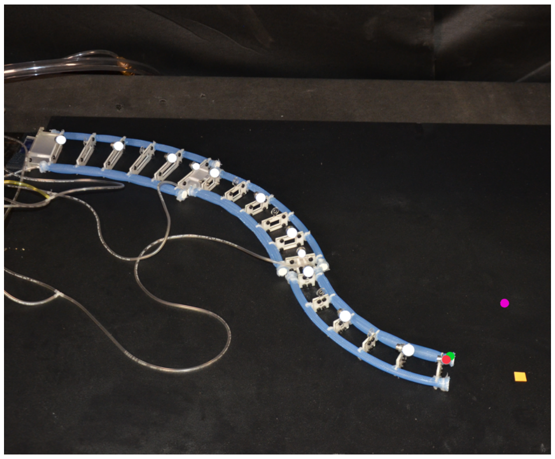
From the table, we can see that the quasistatic model only control performs poorly. This illustrates that without accounting for the system load, the benefits of a staged arm configuration (e.g., increased flexibility near the tip) cannot be taken advantage of. This mode of control performs poorly enough that it would not be suitable for many real-world tasks. As in Experiment 2, we can see that our proposed approach performs closely to that of the curvature control baseline. For all arm configurations, our proposed method achieves an accuracy within 1 cm of the curvature control baseline.



(a) Curvature control



(b) Quasistatic model only control



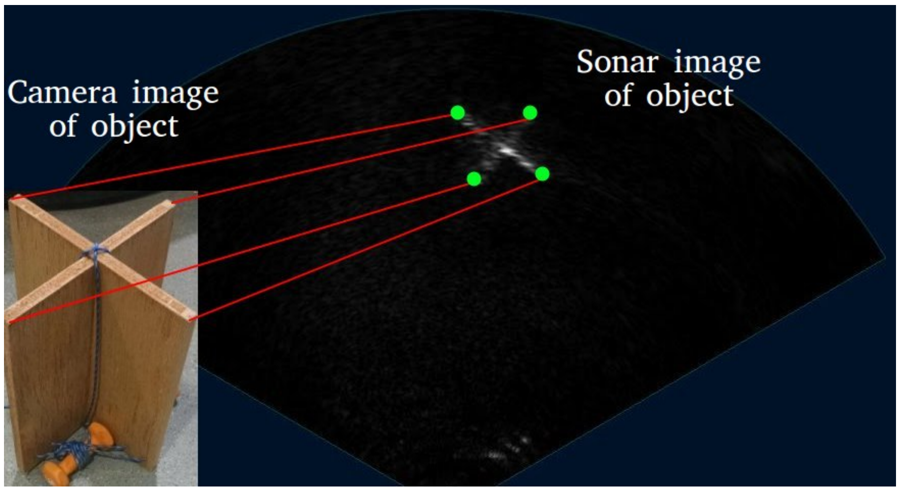
(c) Quasistatic plus load model control

Figure 4.7: A comparison of the end-effector error produced by the control methods used in Experiment 3. In each image, the theoretical expected end-effector location (unreachable due to link shortening) is represented by the orange square. The three dots represent the final end-effector location for the different control methods. Curvature control is represented by the green dot, quasistatic model only control is represented by the magenta dot, and quasistatic plus load model control is represented by the red dot.

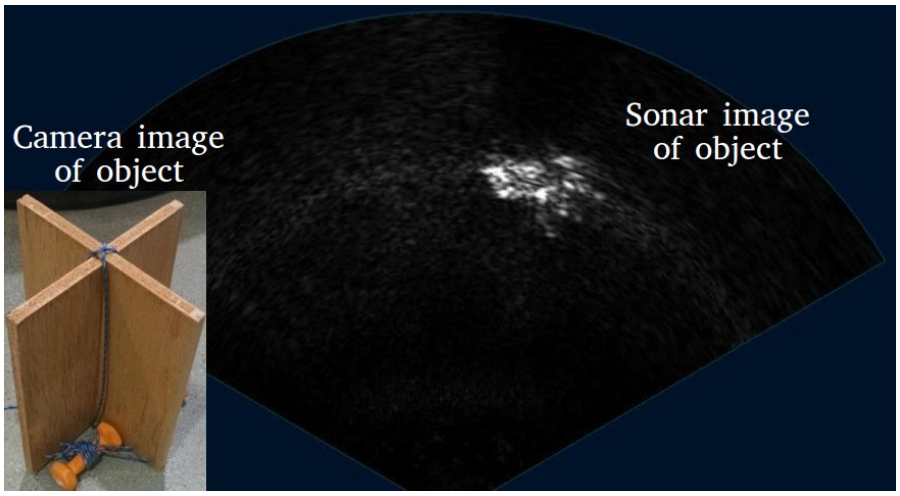
Chapter 5: Sonar Based Underwater Mapping

The 3D reconstruction of underwater environments has proven useful in a variety of applications, including ship hull inspection and underwater surveying tasks [31, 42]. Oftentimes in these and similar applications, 2D imaging sonars are the choice for exteroceptive sensing, as standard cameras have extremely limited visibility in turbid waters. While sonar has superior range, its imagery is often corrupted by noise as well as the non-diffuse reflection of the acoustic wave off of the object of interest [1]. Unlike Chapters 3 and 4, this noise results in data precision so low that automatic feature extractors fail. Given this extremely low precision, Fig. 1.1 suggests incorporating human experts. We further discuss the challenges of sonar noise in the following paragraphs.

Multipath returns are a large producer of noise. Such returns occur when the projected beam completes a sequence of reflections other than simply to the object and back to the sonar (e.g., from the sonar to the seafloor, to the object, and then back to the sonar). Due to the increased length of time for the reflected wave to arrive back at the sonar, these reflections are often imaged at a range beyond the object. This can result in an image similar to Fig. 5.1b, where the horizontal component of the X appears thicker, due to delayed reflections from multipath reflections. Non-diffuse or specular reflection occurs when an object is smooth and fails to reflect the beam back in the direction it was emitted from. An example of this can be seen in Fig. 5.7i, where only the corners of the triangle are shown and the flat top surface does not return energy back to the sonar.



(a) Frame with well-defined features (informative)



(b) Frame with object lacking well-defined features (non-informative)

Figure 5.1: Examples of informative (a human can confidently identify features) and non-informative frames while inspecting a target in the shape of an X. The sub-image in both figures is a camera image of the X shaped target that is insonified. In (a) feature correspondences (red) can be made between the camera image and sonar features (green). In our experiments we found on average only about 39% of the captured frames to be informative.

The substantial amount of noise present in sonar images is problematic for two reasons. First, noise will often corrupt an image so much that analyzing it further would waste time and computational resources. Our method provides a way to spend these resources more efficiently by only providing quality imagery to perception algorithms. Second, noise combined with the low-resolution of sonar imagery hampers the development of an automatic feature extractor for sonar images. For example, feature extractors that use image gradients would incorrectly attempt to extract features from the gradients present in Fig. 5.1b.

The lack of an automatic feature extractor for sonar images necessitates human annotation of the features for 3D reconstructions [32]. Due to the high data rate of imaging sonars, humans cannot annotate the frames in real-time. We show that naively subsampling the data (to allow for real-time annotation) does not result in high-quality reconstructions because noise corrupts many of the images sampled.

The need for human annotation creates a long delay between data collection and environment reconstruction. To eliminate this delay and enable real-time reconstruction, we develop a method to recognize and propose only informative frames to the human for annotation. We define an informative image as one that contains a set of clear and distinguishing features for environment reconstruction (as seen in Fig. 5.1a). This automatic identification, which is done in real-time, allows for a small set of frames to be presented to the operator. This set is small enough to be annotated in real-time.

Unlike the projected LIDAR scans in Chapter 3, in the noisy sonar images the noise is indistinguishable from features on a local scale. We can see that Fig. 1.2 recommends incorporating global context in this case. To identify informative images in noisy

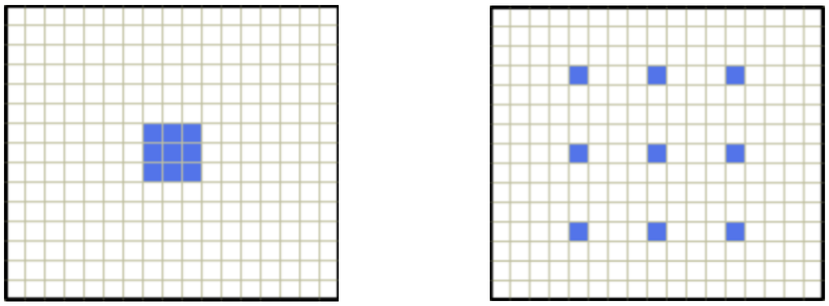

(a) Dilation rate of 1 (b) Dilation rate of 4

Figure 5.2: Example of traditional 3x3 filter (left) and a 3x3 dilated filter (right). The dilated filter uses a larger neighborhood, which compensates for the lack of strong local features in sonar imagery.

and low-resolution sonar imagery we use dilated filters (e.g., Fig. 5.2b) in the atrous convolution architecture, previously used in the computer vision community for image segmentation tasks [100, 12]. Such filters use a large neighborhood of context in analyzing low-resolution and noise-filled sonar images.

In this work we also focus on an approach that is flexible. Given the difficulty in obtaining and labeling sonar imagery, we pay special attention to not overfitting to the objects we train with. This is done by using the context afforded by dilated filters. To demonstrate this, we conduct experiments on imagery of objects not seen in the training set. We thus show that our approach is useful in the mapping or inspecting of objects not recorded in sonar previously. We validate this framework on real sonar imagery containing a variety of objects in different environments.

In summary, we present a novel framework for underwater 3D reconstruction that:

- Automatically selects informative frames for an operator to annotate features, enabling the real-time reconstruction of underwater environments where the features must be manually selected.

- Utilizes the atrous convolution architecture to classify sonar images where features are not well-defined or well-localized.
- Identifies objects not seen in the training set with a higher average precision than previous approaches.

The remainder of this chapter is organized as follows. We first discuss the reconstruction formulation in Section 5.1. Next, we present our proposed approach in Section 5.2. Finally, we describe our experiments and their results in Section 5.3. Versions of the proposed approach and results presented in this chapter have appeared in our prior workshop paper [17] and conference paper [18].

5.1 Reconstruction Formulation

We formulate the ASFM problem using a factor graph as proposed by Huang and Kaess [32]. An overview of the process is provided here, with a graphical representation shown in Fig. 5.3. For full details, please refer to [32, 44].

In formulating the reconstruction objective, we seek to find the maximal probability set of poses, $\Theta = \{b_i, l_j\}$, and observations, $Z = \{u_i, m_k\}$, where b_i is a robot pose, l_j is a landmark, u_i is a navigation measurement, and m_k is a landmark measurement. Assuming Gaussian noise, this can be formulated as a nonlinear least-squares problem:

$$\Theta^* = \underset{\Theta,Z}{\operatorname{argmin}}[\ \|b_0\|^2_\Lambda + \sum_{k=1}^{M} \|h(b_i, l_j) - m_k\|^2_{\Xi_k} + \sum_{i=1}^{N} \|g(b_i, b_{i-1}) - u_i\|^2_{\Lambda_i}\ ], \tag{5.1}$$

where $\|b\|^2_\Sigma = b^T\Sigma^{-1}b$ is the Mahalanobis distance squared and *M* and *N* are the number of sensor and odometry measurements respectively. The sensor model is defined as $h(b_i, l_j) + \mathcal{N}(0, \Xi_k)$ and the odometry model is defined as $g(b_i, b_{i-1}) + \mathcal{N}(0, \Lambda_i)$, where Ξ_k is the variance of sensor measurement k and Λ_i is the variance in odometry measurement i.

In this formulation, there are six unknowns $(x, y, z, \alpha, \beta, \gamma)$ for every pose and three unknowns (x, y, z) for every landmark where α, β, and γ are the roll, pitch, and yaw

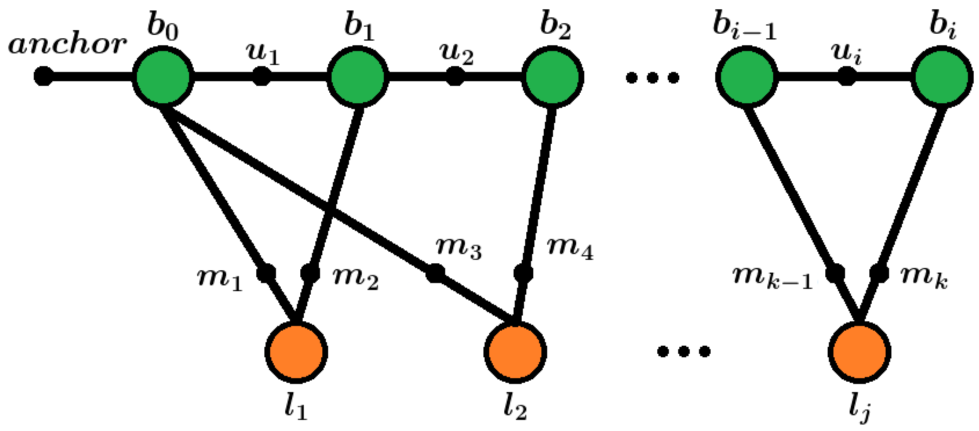


Figure 5.3: Factor graph representation. l, m, b, and u are the landmark positions, landmark measurements, robot pose, and navigation measurements respectively. The landmarks are hand-labeled features in sonar images (green dots in Fig. 5.1a).

respectively. With all landmarks visible from every pose, fixing the first frame yields a fully constrained system iff: $6(n-1)+3m \leq 2mn$, where n is the number of robot poses and m is the number of landmarks in the factor graph. Rearranging this, we derive n, the number of frames necessary for reconstruction, to be

$$n \geq \frac{3m-6}{2m-6}. \tag{5.2}$$

While this formulation is well-suited to the reconstruction process, it does not directly support performing these reconstructions in real-time. In developing such a solution, we introduce an image proposal system which gives a human operator a small set of frames containing information useful for reconstruction.

5.2 Methodology

In this section, we detail the different components of our proposed approach. In Section 5.2.1 we discuss how we select informative frames. Next, we present our selected network architecture in Section 5.2.2. We then discuss the frame proposal process and threshold tuning in Sections 5.2.3 and 5.2.4, respectively. Finally, we highlight how we perform the reconstruction process in Section 5.2.5.

5.2.1 Proposing Informative Frames

To select informative frames automatically we leverage the atrous convolution architecture. We motivate and briefly summarize our approach here.

Due to the lack of strong local features in sonar images, the CNN method of analyzing sonar imagery can be greatly improved by using dilated filters (shown in Fig. 5.2b). Dilated filters use the same number of pixels as a standard filter; however because they are distributed, they can alleviate the effects of noise and low-resolution in sonar images.

The ability to ignore and not overfit to local features also improves the transfer learning capabilities in identifying informative sonar images. That is, given an image of an object not seen in training data, the atrous CNN is able to classify informative images with a higher average precision than a standard CNN or frequency-component based method. This ability is particularly attractive when using an underwater sonar, where there is a lack of training data as compared with terrestrial environments, and various sources of noise make similar looking objects appear different in sonar imagery.

5.2.2 Architecture Choice

To determine the appropriate architecture for configuring these dilated filters for use in analyzing sonar imagery, we test multiple architecture configurations and parameter choices. Full details of our selection process can be found in our prior work; a summary is below [17].

For the general architecture choice we evaluated three approaches. The first is similar to a normal CNN except the convolutional layers beyond the first have dilated filters instead of standard filters. This is inspired by Yu and Koltun, who use this configuration of dilated filters for dense prediction in image segmentation tasks [100]. The second pretrains two networks: one a standard CNN, the other an atrous network. The dense layers are then popped off of each and combined into a single dense layer for training again. This is meant to capture the expressibility of CNNs with the generalization capabilities of atrous networks. Finally, we test an architecture that uses filters of different dilation rates in the same layer. This is inspired by Chen et al. who use this configuration for semantic image segmentation [12]. We complete a full parameter sweep on each architecture, including number of layers, number of filters, kernel size, dilation rate, and dense layer size. Average precision was used as the metric for comparing each architecture and parameter configuration. To account for variability in initialization, the results for each were averaged over 20 runs. We found the architecture in Fig. 5.4 to achieve the highest average precision. Due to the large dilation rate, this network starts to extract non-local features as early as the second convolutional layer. Therefore, it does not overfit to local patterns in the training set and has better generalization capabilities.

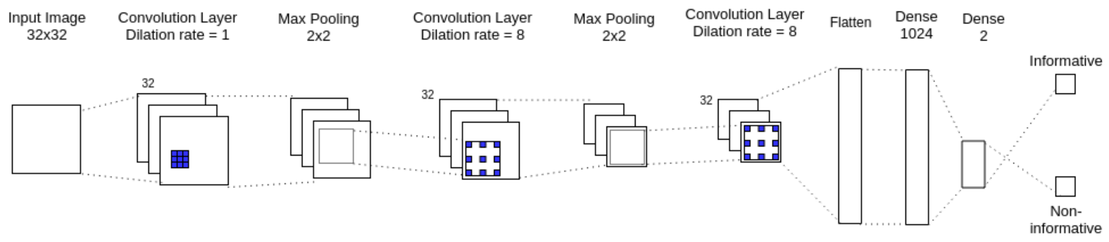


Figure 5.4: The atrous convolutional network used. The parameters and filter configuration were tuned by testing discrete options.

5.2.3 Frame Proposal Process

We treat the real-time frame proposal problem as a single-shot information-greedy selection. That is, every time a frame is proposed to the human operator, the most informative frame is selected from the set of candidate frames, $max(n_{info}) \in N$. The informativeness, n_{info}, of a frame is determined directly from the sigmoid output of our atrous convolution based deep network [17].

The set of candidate frames at any given time, N, is determined by two factors: n_{info} and the time since the last frame proposal, t_{motion}. Frame n_i is added to the set of candidate frames, $N \cup \{n_i\}$ if:

$$\begin{aligned} n_{info} &\geq T_{info} \\ t_{motion} &\geq T_{motion}. \end{aligned} \tag{5.3}$$

The threshold T_{info} allows for the tradeoff between frame information quality and frame quantity. The threshold T_{motion} is tuned to allow for view diversity while considering potential vehicle motion and sonar field-of-view.

Frames are proposed to the human operator sequentially as soon as the labeling of the previous frame is complete. When a frame is proposed, the set of candidate frames, N, is cleared. Frame proposal stops when either an appropriate number of frames, N_{thresh}, have been labeled or the data stream ends (e.g., the deployment concludes).

Once a frame is proposed, the human operator selects the features to be reconstructed by simply clicking on the image. We found this usually involved a small number of features (3-6), which allowed for fast feature selection and reconstruction. An example of selected features for the "X" shaped object are shown as green dots in Fig. 5.1a. An interesting avenue for future work is the investigation of the effects of the experience or training of the human labeler. We note that given the simplicity of the task, we do not expect large variations across annotators.

5.2.4 Threshold Tuning

We set the threshold for proposal, T_{info} at 0.99. As demonstrated in Eq. 5.2, with objects having on the order of five features, often less than ten frames are needed to complete reconstructions. In these cases it is more beneficial to be selective in choosing frames (high threshold) rather than finding many informative frames. Experimentally this threshold provided more than enough frames to complete the reconstruction. For completeness, the precision recall curve from our model on validation data is shown in Fig. 5.5.

To determine N_{thresh}, we evaluated the reconstruction error for annotated set sizes of n (the minimum number of frames for reconstruction) to $2n + 1$. To obtain the average and variance on the sum squared reconstruction error (SSE) we use a set of data containing 7 proposed frames. From this set we test all of the combinations of frames for each annotated set size. These combinations are given by $\binom{2n+1}{k}$ where $2n + 1 = 7$ and $k = \{3, 4, 5, 6\}$. The same validation data used in completing the parameter sweep for the atrous network was used. As seen in Fig. 5.8, we find $N_{thresh} = 5$ to provide enough frames for a low reconstruction error/variance, with minor benefits increasing the set size to 6.

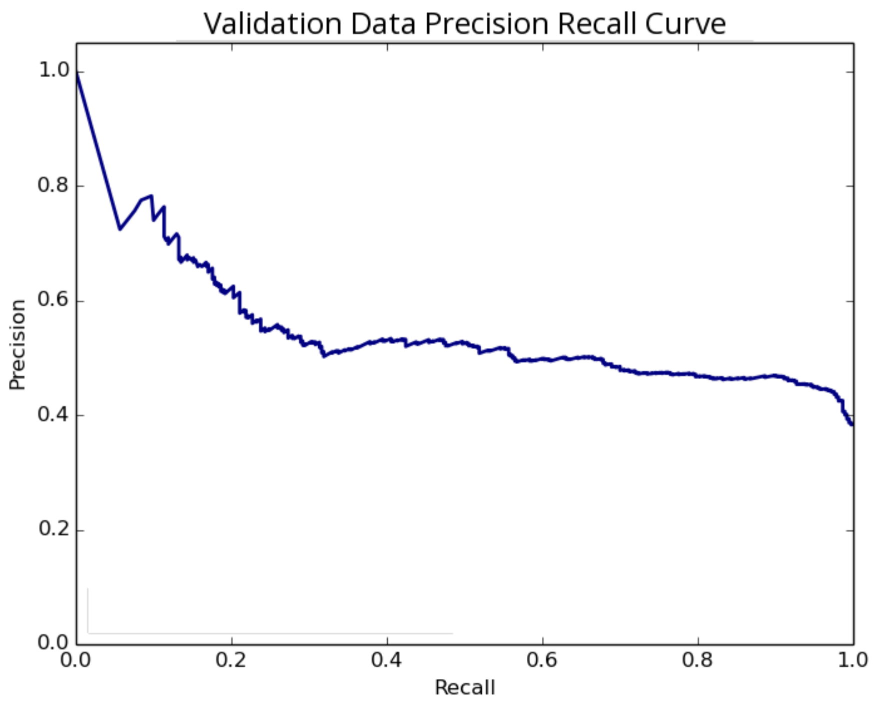


Figure 5.5: The precision recall curve generated from validation data used to choose a threshold for informative vs. non-informative frames, denoted as T_{info}.

For the experiments we set $T_{motion} = 2$ sec. This threshold provided view diversity while allowing the objects of interest to remain in the sonar field-of-view. It also allowed the human operator to annotate each frame in real-time.

5.2.5 Reconstruction Process

Inspired by Huang and Kaess, initial landmark locations in the factor graph are calculated by setting the elevation angle ϕ to 0° and projecting the sonar feature points (r, θ, ϕ) into Euclidean coordinates (X, Y, Z) [32]. The nonlinear least-squares problem is then optimized via iterative linearization using the Levenberg-Marquardt (LM) algorithm.

5.3 Experiments and Results

To demonstrate the capability of our framework to complete accurate reconstructions in real-time, we ran three experiments on real sonar data played back offline. The first shows the transfer learning capabilities of our atrous network when tested on imagery of objects not seen in training. In the second experiment, we show that we select an appropriate number of frames for our reconstruction. That is, we choose enough to fully constrain the optimization but do not propose many superfluous frames. Finally, the third experiment shows that our proposal method chooses a subset of frames that enable real-time reconstructions with errors lower than baseline methods.

In the first experiment we test using 4000 frames of the *Multi* dataset (the remaining 1000 frames were used as validation data). In Experiment 2 we use these same frames as well as the *Cinder* dataset. In Experiment 3, we only use data from the *Cinder* dataset. Unless stated otherwise, test data for each experiment contains imagery of objects not trained on, thus demonstrating the flexibility or lack thereof of each approach examined. This also serves to demonstrate the performance of our approach when trained prior to deployments.

Throughout each test we maintain a real-time property by ensuring that at the end of a given experiment, our algorithm has output the reconstructed shape. Experiments 1 and 2 are used only for analyzing components of our process and are thus not timed.

Experiment 3 mimics a deployment and thus in Section 5.3.5 we provide runtime from start to finish of our process. For completeness we provide average numbers for each component of our system. Our CNN processes images in 18.1 ± 0.8 ms (55.2 ± 2.41 Hz). We found that a human operator annotated images of the X object in 2.63 ± 0.231 sec. Finally, the factor graph bundle adjustment process took on average 77.1 ± 9.4 ms.

5.3.1 System Overview

For our experiments we use a Tritech Gemini 720i multi-beam sonar onboard a tethered Seabotix vLBV300 Remotely Operated Vehicle (shown in Fig. 5.6).

The Gemini 720i is a standard multi-beam imaging sonar with similar parameters to the SoundMetrics DIDSON, BlueView M900-90, and the Aris Explorer 3000, all of which have been used in previous work [32, 55, 38]. It operates at 720kHz, images a 120° swath horizontally, has a maximum range of 120m, and produces images at 20Hz. This data rate, combined with large amounts of image noise, creates a situation in which the human operator is presented with a large number of frames, many of which will not be useful.

Our vehicle is also equipped with a Doppler Velocity Log and Inertial Navigation System which we use for pose estimates in the factor graph mapping scheme. The vehicle's tether provides continual power as well as data transmission back to an offboard computer which can be used to analyze sonar imagery in real-time. The use of standard underwater sensors demonstrates our approach can be deployed on a variety of platforms for real-time underwater reconstructions.

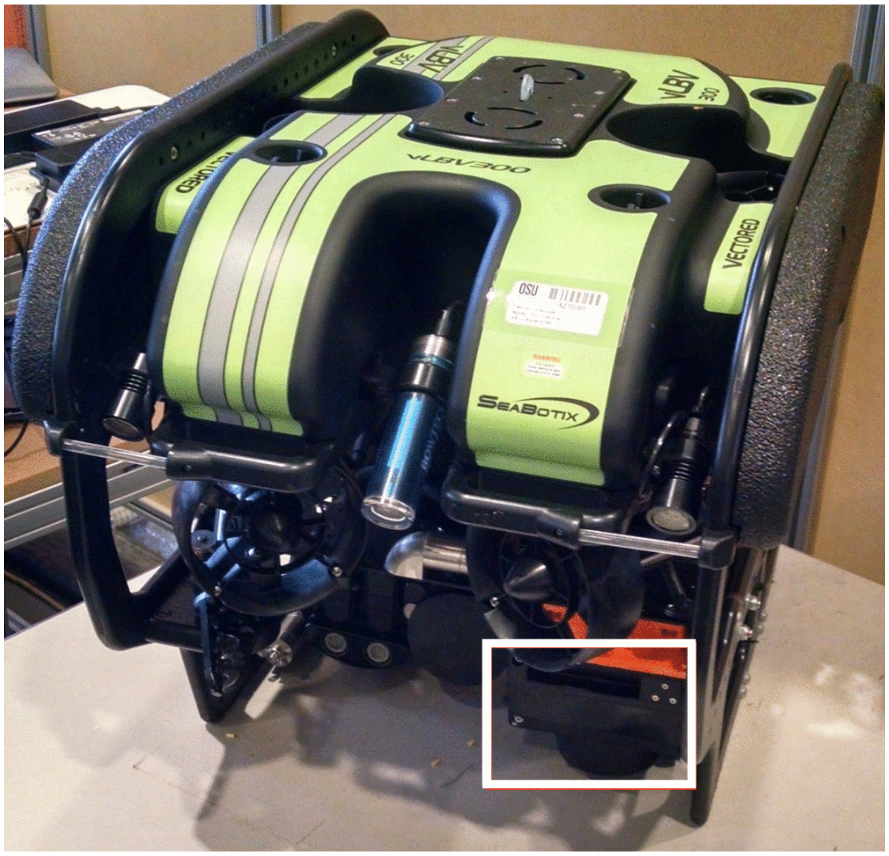


Figure 5.6: The Seabotix vLBV300 and Gemini 720i imaging sonar (designated by the white rectangle in the lower right).

5.3.2 Dataset Overview

In this work we use four datasets captured onboard our vehicle while it operated in a variety of environments. Each dataset was collected in a passive manner, meaning each dataset is a single contiguous video stream captured while our vehicle moved through underwater environments. The content and number of informative frames varied across datasets and is summarized in Table 5.1.

In dataset X_1 we collect a set of 5000 frames of the X shaped object (Fig. 5.7a) in

Table 5.1: Summary of the datasets

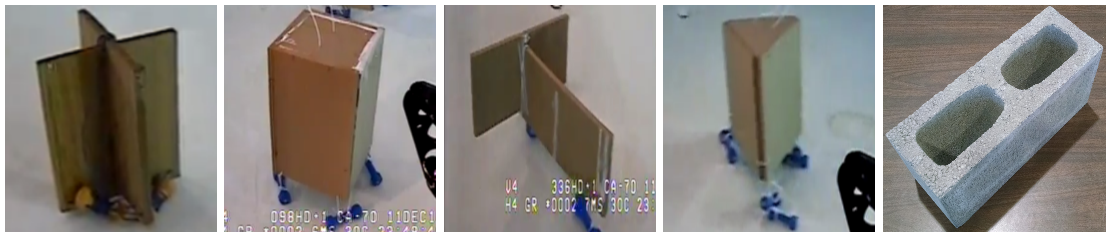
Dataset	Total number of frames	Percent informative frames
X_1	5000	36%
X_2	1107	56%
$Multi$	5000	39%
$Cinder$	1470	42%

Note: An informative frame is one in which a human operator can clearly identify an object and its features in the sonar image.

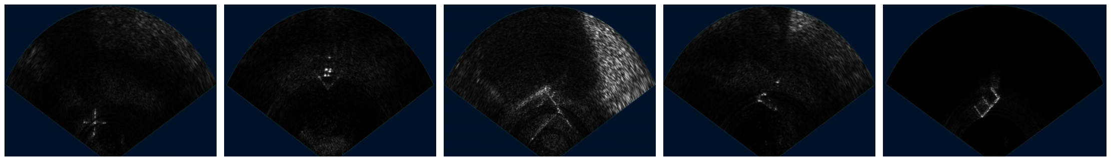
the Oregon State University pool. In dataset X_2 we collect 1107 frames of the same X object in the same environment on a different date. In dataset $Multi$ we collect a set of 5000 frames containing insonified imagery of the X, Square, Triangle, and T-shaped objects (Figs. 5.7a-5.7d) in the pool. This dataset contains images with just a single object in addition to images containing multiple objects. Finally, in dataset $Cinder$ we capture 1470 frames of a cinder block target (Fig. 5.7e) in Yaquina Bay, Newport, OR. While in this work we use representative shapes for the underwater domain (resembling objects such as underwater moorings) an interesting avenue for future work involves the use of our architecture on more complex shapes.

The binary ground truth label (informative or non-informative) is obtained by having an expert evaluate if each image contains enough well-defined features to clearly identify the object. For example, simply seeing two lines intersect does not identify one of the five targets; however observing three well defined lines that each meet in an acute fashion clearly defines the triangle object seen in Fig. 5.7d.

Throughout the three experiments we train primarily using datasets X_1 and X_2 (both only containing data of the X target in Fig. 5.7a). 1000 frames of the $Multi$ dataset are used as validation data for the tuning of parameters.



(a) X target (b) Square target (c) T target (d) Triangle target (e) Cinder block target



(f) X in sonar space (g) Square in sonar space (h) T in sonar space (i) Triangle in sonar space (j) Cinder block in sonar space

Figure 5.7: Each target and its representation in sonar image space. (a)-(d) are images taken from the vehicle underwater. As a note, in (j) the second long side of the cinder block has been occluded.

5.3.3 Experiment 1: Validating Transfer Learning Capabilities

In the first experiment we show that our model is able to identify objects not in the training set with a higher average precision than baseline methods. We are able to do this while also classifying objects seen in the training set with precision comparable to baseline methods. This ability is a result of our atrous network, which balances the power of standard convolutional architectures with the more global context afforded by dilated filters.

We compare against three baselines. The first is the Discrete Cosine Transform (DCT) method of global analysis. Defined for pixels m, n of an image A of size $M \times N$

as:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} cos \frac{\pi(2m+1)p}{2M} cos \frac{\pi(2n+1)q}{2N}, \tag{5.4}$$

where

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, p = 0 \\ \sqrt{\frac{2}{M}}, 1 \leq p \leq M-1 \end{cases} \tag{5.5}$$

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, q = 0 \\ \sqrt{\frac{2}{N}}, 1 \leq q \leq N-1\,, \end{cases} \tag{5.6}$$

and B_{pq} is the coefficient on the basis cosine function defined by the constant p and q [61]. The resulting coefficients represent the frequency components contained in the image, which can be used to separate noisy images from those with more complex geometries. We complete this classification with the use of a Support Vector Machine using the frequency components as the input data. This method is natural for sonar imagery because it takes a global image analysis approach to imagery with poorly-defined features. The second baseline is the approach taken by Kim et al. (called "two-layer CNN") [47]. The third and final baseline is a network we developed inspired by Kim et al. that contains an additional convolutional layer (called "three-layer CNN").

We compare the four approaches in three scenarios, the results of which are shown in Table 5.2. The first scenario allows each classifier to train with 600 frames of multi-object data (called "seeding"). The test data contains imagery of each of the four objects in Figs. 5.7a-5.7d (including the X). These results show that the DCT method of global analysis is not expressive enough to capture the features present in both the training and

Table 5.2: Average precision for baselines and our method

Method	Avg. Precision		
	Seeding X present	Seeding No X	No Seeding No X
DCT [61]	0.57±0.01	0.55±0.02	0.48±0.02
Two-layer CNN [47]	0.69±0.04	0.58±0.05	0.42±0.04
Three-layer CNN	0.70±0.05	0.64±0.04	0.42±0.03
Atrous Convolution (ours)	**0.72±0.02**	**0.68±0.03**	**0.54±0.04**

test data. The two and three-layer CNNs, as well as our method, all perform similarly.

The second scenario still seeds the training of the models, but the test data no longer contains imagery of the X object. We find that our model outperforms the others, which is expected because dilated filters capture a larger neighborhood of influence than localized filters.

Finally, in the third scenario, true transfer learning is evaluated. The models are not given imagery of any object but the X to train on and the test set only contains imagery of the Triangle, Square, and T shaped objects. In this scenario we find that our model significantly outperforms other methods, which is intuitive as dilated filters give a nice balance between the power of deep learning based methods and the larger context of global evaluation methods. As demonstrated in the first scenario, this does not come at the cost of performing worse than baselines in the case where the test data is similar to the training data.

5.3.4 Experiment 2: Number of Frames to Propose

In the second experiment, we show that our method proposes an appropriate number of frames. That is, our method achieves lower reconstruction error than using the minimum number of frames to fully constrain the reconstruction (n) as well as achieving comparable performance to reconstructions that use a large number of frames ($2n$). Given the lack of global ground truth coordinates, known object dimensions are used to compute the reconstruction error.

The results of this experiment can be seen in Fig. 5.8. As expected, increasing the number of frames used in the reconstruction decreases the reconstruction error. When only using the minimum number of frames required, n, we can see that on average the reconstruction has both the highest sum squared error (SSE) as well as the highest variance. At five frames in both plots we observe a “knee” point, where increasing the size to six gives only marginal benefit.

5.3.5 Experiment 3: Selecting Informative Subsets

In the third experiment, we demonstrate the ability for our method to select informative frames and operate more efficiently than standard baselines. For this experiment we use a subset of 400 contiguous frames. The results of this experiment can be seen in Table 5.3. Statistical bounds have been provided for the comparison to [33] due to the random nature of frame proposals in this baseline. These bounds were found averaging over 5 runs.

The first baseline we compare against is naively presenting the user with every frame

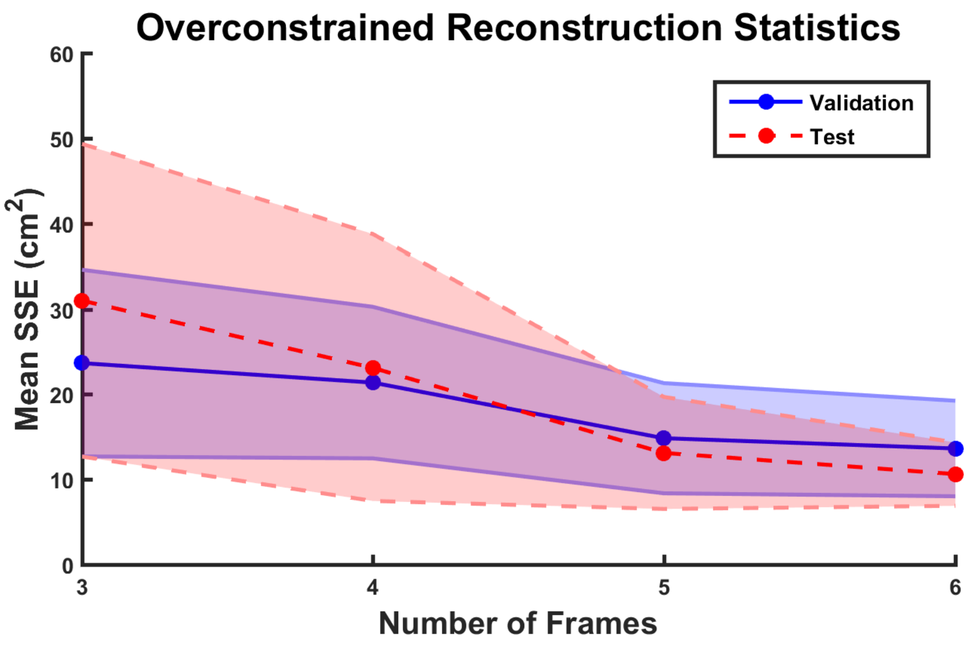


Figure 5.8: Selection of the appropriate number of frames to use for reconstructions. For both validation data (blue line) as well as test data (red line) five frames is at a “knee” point as increasing to six frames gives only a marginal benefit.

Table 5.3: Evaluation of our reconstruction framework

Method	**Number of frames proposed**	**Reconstruction SSE** (cm^2)
Every frame	400	8761.9
Every informative frame	167	16.37
Naive subsampling	**5**	71.239
Huang and Kaess [33]	6.93±1.91	18.42±6.47
Atrous proposal system (ours)	**5**	**13.12±3.72**

for annotation. We note that this method requires the human operator to annotate every frame and thus cannot be completed in real-time. The large amount of error is due to the fact that many poor quality frames are annotated by the user. Such large error demonstrates the low-quality of many of the images, which do not contain enough clear features to be annotated by the human.

The second baseline we test is proposing every informative frame to the user for annotation. The informativeness of a frame is determined by the sigmoid output of our atrous CNN. This baseline achieves a reconstruction error similar to that of our method; however we note that it cannot be run in real-time. The low reconstruction error is expected because the labeled images are all informative frames.

In the third baseline, we naively subsample all frames at an even interval to only present the human operator with the same number of frames as our method. This is equivalent to naively reducing the data rate to allow for real-time labeling. While this allows for real-time performance, we can see that the reconstruction error achieved is poor. This is due to the fact that without first assessing a frame's informativeness, poor quality frames are still presented to the user.

The fourth baseline allows us to compare against a method inspired by the state-of-the-art in underwater reconstruction [33]. In this previous work, Huang and Kaess perform automatic data association between annotated frames to determine whether or not all annotated feature points can be associated. If they cannot, the frame is rejected and not used for reconstruction purposes. To extend this approach and enable real-time reconstructions, we randomly select frames to propose to the human operator. If an association exists between the annotated feature points, the frame is used in the reconstruction. If not, it is discarded and another is proposed. The proposal process runs until the minimum number of frames required, n, have been successfully annotated. While this method is able to achieve high quality reconstructions, the reconstruction error is both higher and more variable than our method. This is due to the fact that only the minimum number of frames required are used in the reconstruction process. We also note that since frames are not evaluated before proposal, this baseline typically proposes a larger number of frames than our method.

The last row of the table presents the results of our method. We note that we achieve the lowest average reconstruction error and variability, while simultaneously proposing the lowest number of sonar frames to the user. This low reconstruction error is achieved while using N_{thresh} = 5 frames. We also maintain real-time performance during the 20 second experiment. Our method, from start to finish, took on average 14.88 ± 0.49 seconds to complete.

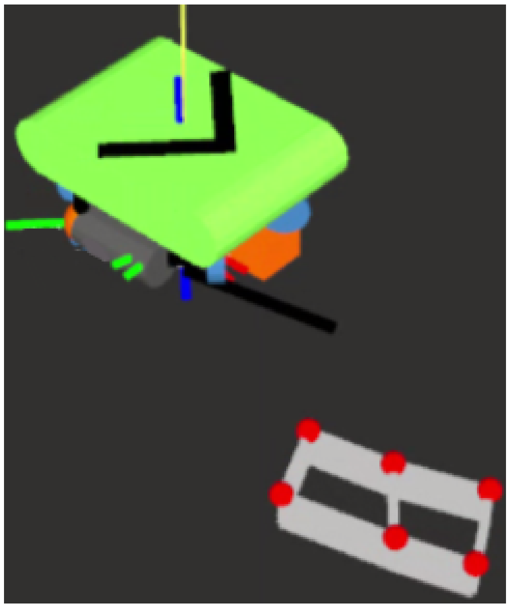


Figure 5.9: The vehicle and reconstruction of the cinder block in the RViz simulation environment. Our method outputs the red dots (chosen as features by the operator) and the gray section was extrapolated.

Chapter 6: Conclusion

In this work, we proposed augmenting the power of deep learning techniques with tools such as optimization methods, physics based models, and human expertise as a means to circumvent the challenges associated with end-to-end deep learning approaches. In doing so, we presented a principled framework for approaching problems in the field of robotics that allows a user to identify the types of augmentation methods and deep learning techniques best suited to their problem. We validated our framework in three different domains: LIDAR based odometry estimation, hybrid soft robotic control, and sonar based underwater mapping (Fig. 1.1 and 1.2).

First, we investigated LIDAR based odometry estimation which can be characterized with both high data precision and availability due to the Velodyne sensors used. This is ideal for augmenting with optimization methods. In image space, the LIDAR features are locally correlated. We presented a novel pre-processing step that enables faster, and in the case of standard techniques, more accurate scan matching for point clouds generated by modern LIDARs. We enable this by leveraging the robust feature extraction capabilities of convolutional neural networks and denoising autoencoders. Through experiments on multiple real-world datasets, we showed the features learned to be both robust and generalizable. While bootstrapping the quality of standard techniques is useful, we found the potential speed increases with state-of-the-art techniques to be of particular interest and an interesting avenue to explore further.

Next, we examined hybrid soft robotic control which has lower data precision due to real-world noise (e.g., friction and manufacturing imperfections). Here, augmenting with model based methods is more appropriate. Additionally, data availability is restricted due to the time consuming nature of data collection. During the classification stage of our approach, data is temporally correlated while it is not during the modeling stage. Our presented approach is capable of learning to model the system load introduced when constructing tapered staged soft arm configurations that is not accounted for in the generalized kinematic model. Additionally, it is capable of modeling multiple load states (and classifying between them) allowing for the staged arms to be reconfigured with no additional training required. Through experiments on soft robotic hardware, we showed that our method is capable of classifying between different arm configurations at a rate greater than 95%. Additionally, our method is capable of reducing the end-effector error of quasistatic model only control to within 1 cm of our controller baseline. Our approach enables the use of these staged arm configurations allowing us to take advantage of the increased flexibility provided by the increasingly reduced link segment widths.

Finally, we considered the sonar based underwater mapping domain. Here, the data is so noisy that augmenting with human experts and some global context is required. While we again collect our own data here, the high data rate of the Gemini imaging sonar provides a large amount of data. We presented a novel framework that enables the capability to complete accurate underwater 3D reconstructions in real-time and overcoming the large amounts of noise present in sonar imagery. We enable this functionality by identifying a small subset of informative frames for the human to annotate.

Through experiments on real sonar images we showed our atrous network outperformed other classifiers in identifying informative frames for proposal. We also experimentally validated the ability of our network to leverage non-local features to complete transfer learning.

Chapter 7: Future Work

Through our work in three different domains, we found many interesting avenues for future research. While out of scope for this work, they may provide opportunities to others working in these areas. In this chapter, we provide a brief overview of possible extensions to the work presented in this dissertation and speculate on how our framework could be applied to other domains.

7.1 LIDAR Based Odometry Estimation

In this domain, additional avenues for future work involve further generalization and more sophisticated network structures. Preliminary testing indicates this method may be suitable for RGB-D data; however it is not yet clear what changes (if any) will optimize performance. Additional network considerations include adaptive discretization, incorporating intensity measurements, and direct point cloud convolution.

7.2 Hybrid Soft Robotic Control

Our results in this domain suggest several avenues for future work. The first is to further extend the arm configuration classification step to predict a segment's general load state. This would enable the arm to not only work in various configurations, but with external loads as well (e.g., manipulating an object). In this scenario, we no longer have distinct load states from well defined arm configurations. As such, we would likely wish to update the proposed architecture to directly input the LSTM's latent representation into the fully connected network, allowing for it to better learn over the full load space. A limiting factor is the current strength of the pneumatic actuators. Currently, the base links require most of the available force to simply move the staged arms.

A second avenue for future work is to further increase the real-world capability of the staged arms. This could be done by incorporating soft sensors on the individual link segments, removing the need for the OptiTrack system entirely. Feedback from these sensors, when coupled with the pressure state, could allow for real-time load predictions (e.g., during and after manipulating an object) as well as detecting and exploiting collision states.

7.3 Sonar Based Underwater Mapping

One particularly interesting extension to this work in this domain is the formal treatment of incorporating image diversity into our framework. While in this work we enforce diversity with T_{motion}, the effects of using diverse images into the reconstruction process remains an interesting avenue to explore. Additionally, exploring methods for learning to identify the unknown elevation angle directly from sonar image inspection could improve the reconstruction process.

7.4 Other Domains

Other interesting domains to investigate include robotic manipulation, ocean monitoring, and human-like motion planning. In this section, we discuss considerations for how these domains might fit into the framework presented. An illustration of where they might fit can be seen in Figs. 7.1 and 7.2.

In regards to robotic manipulation, a robotic arm is likely to have a sensor capable of providing both image and depth data. Such RGB-D sensors typically have high

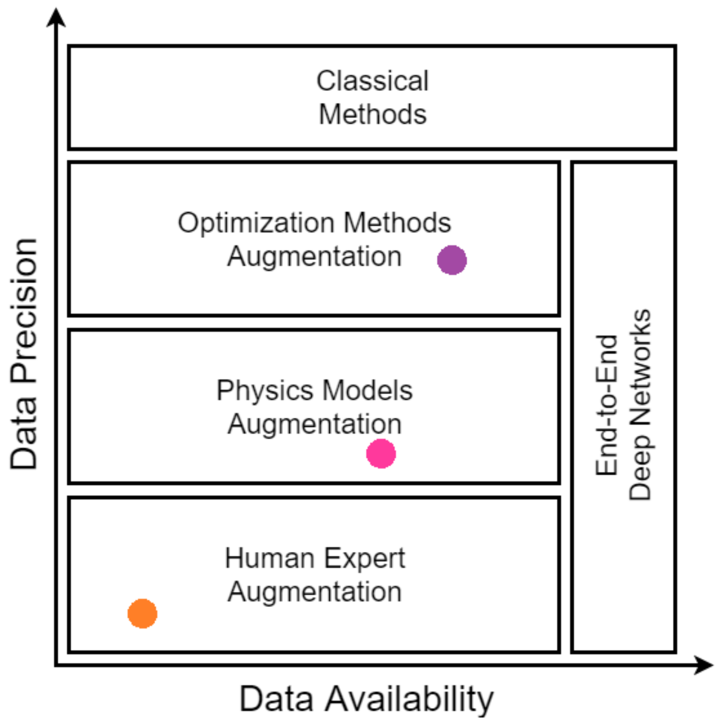

Classical Methods
Optimization Methods Augmentation
Physics Models Augmentation
Human Expert Augmentation
End-to-End Deep Networks
Data Precision
Data Availability

Figure 7.1: A visualize representation of the data space and various augmentation methods. The colored dots represent two possible domains to investigate. Purple represents robotic manipulation, pink represents ocean monitoring, and orange represents human-like motion planning.

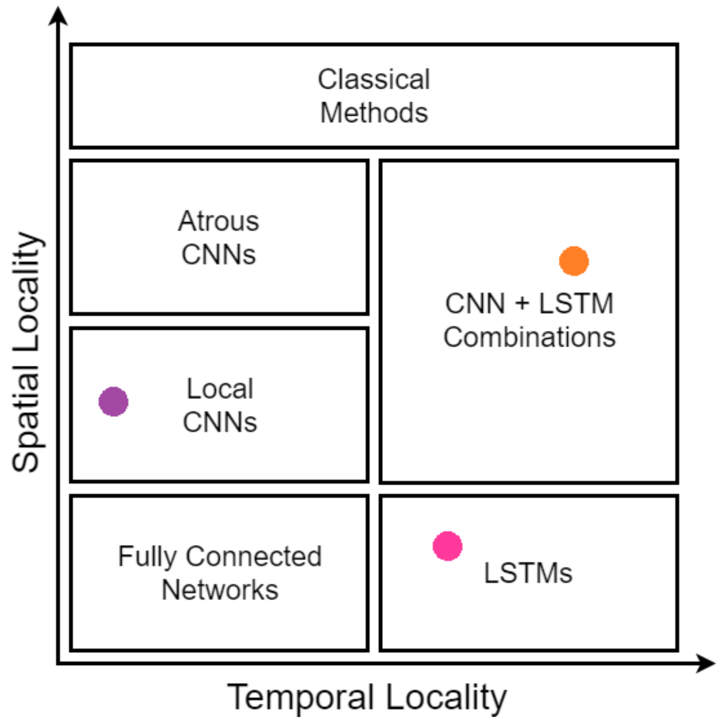


Figure 7.2: A visualize representation of the data space and various augmentation methods. The colored dots represent two possible domains to investigate. Purple represents robotic manipulation, pink represents ocean monitoring, and orange represents human-like motion planning.

precision, although less than full 3D LIDARs such as Velodynes. The high data rate associated with these sensors would allow for high data availability. While cost prohibitive, with enough sets of hardware, the data availability may even be high enough to use end-to-end deep networks. As such, augmenting with optimization methods makes sense. Images provided from the sensor can be used to classify and localize the item in the scene. Given a model of a desired item to manipulate, optimization based scan matching methods could be used to further refine the prediction and pose of the object. In both cases, features should be locally correlated and CNNs would be useful.

In the ocean monitoring domain, we expect our data to be less precise than in the robotic manipulation domain. Sensor measurements are more likely to be noisy and less frequent. In this case, our framework suggests that augmenting with physics models may be appropriate. In doing so, a deployed robot could model environmental factors such as ocean currents to aid in navigation. Data recorded by the onboard sensors (e.g., temperature) is likely to be correlated temporally and LSTMs could leverage this. Measurements recorded as the robot navigates could provide data in various locations. However, it is unlikely that the robot would visit a wide enough variety of locations to benefit from CNN based approaches.

Human-like motion planning is a more difficult problem. Paths generated through planning algorithms tend to optimize for distance or cost rather than various other factors that humans inherently account for. As such, this data is likely to not be very accurate or precise. Instead, human experts could be leveraged to annotate waypoints (or paths) in a scene. However, incorporating a human component will result in the data availability being lower. Locally correlated features would be useful in analyzing the structure of a map. Depending on the maps considered, it may turn out to be useful to incorporate some global context as well. In this case, the temporal aspect of waypoint ordering will likely benefit from the use of LSTMs.

Bibliography

[1] M. D. Aykin and S. S. Negahdaripour. Modeling 2-d lens-based forward-scan sonar imagery for targets with diffuse reflectance. *IEEE Journal of Oceanic Engineering*, 41(3):569–582, 2016.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Proc. Advances in Neural Information Processing Systems*, pages 153–160, 2007.

[3] P. Bergström and O. Edlund. Robust registration of point sets using iteratively reweighted least squares. *Computational Optimization and Applications*, 58(3):543–561, 2014.

[4] P. J. Besl and N. D. McKay. Method for registration of 3-D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[5] M. Bosse and R. Zlot. Continuous 3D scan-matching with a spinning 2D laser. In *Proc. IEEE International Conference on Robotics and Automation*, pages 4312–4319, 2009.

[6] C. Branyan, C. Fleming, J. Remaley, A. Kothari, K. Tumer, R. Hatton, and Y. Menguc. Soft snake robots: Mechanical design and geometric gait implementation. In *Proc. IEEE Conference on Robotics and Biomimetics*, pages 282–289, 2017.

[7] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan. Modeling and control of soft robots using the koopman operator and model predictive control. In *Proc. Robotics: Science and Systems Conference*, Freiburg, Germany, 2019.

[8] M. Calisti, E. Falotico, and C. Laschi. Hopping on uneven terrains with an underwater one-legged robot. *IEEE Robotics and Automation Letters*, 1:461–468, 2016.

[9] D. B. Camarillo, C. R. Carlson, and J. K. Salisbury. Configuration tracking for continuum manipulators with coupled tendon drive. *IEEE Transactions on Robotics*, 25(4):798–808, 2009.

[10] J. Campbell, R. Sukthankar, and I. Nourbakhsh. Techniques for evaluating optical flow for visual odometry in extreme terrain. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3704–3711, 2004.

[11] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice. University of michigan north campus long-term vision and LIDAR dataset. *The International Journal of Robotics Research*, 35(9):1023–1035, 2016.

[12] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. In *Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 834–848, 2017.

[13] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2724–2729, 1991.

[14] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

[15] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[16] J. M. Croom, D. C. Rucker, J. M. Romano, and R. J. Webster. Visual sensing of continuum robot shape using self-organizing maps. In *Proc. IEEE International Conference on Robotics and Automation*, pages 4591–4596, 2010.

[17] R. DeBortoli, A. Nicolai, F. Li, and G. A. Hollinger. Assessing perception quality in sonar images using global context. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Introspective Methods for Reliable Autonomy*, Vancouver, Canada, 2017.

[18] R. DeBortoli, A. Nicolai, F. Li, and G. A. Hollinger. Real-time underwater 3d reconstruction using global context and active labeling. In *Proc. IEEE International Conference on Robotics and Automation*, pages 6204–6211, 2018.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, Florida, 2009.

[20] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. de Fauw, M. Heilman, D. M.

de Almeida, B. McFee, H. Weideman, G. Takács, P. de Rivaz, J. Crall, G. Sanders, K. Rasul, C. Liu, G. French, and J. Degrave. Lasagne: First release. `http://dx.doi.org/10.5281/zenodo.27878`, Aug 2015.

[21] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[22] J. Garstka and G. Peters. Fast and robust keypoint detection in unstructured 3-D point clouds. In *Proc. International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 131–140, 2015.

[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[24] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi. Learning the inverse kinetics of an octopus-like manipulator in three-dimensional space. *Bioinspiration & Biomimetics*, 10(3):035006, May 2015.

[25] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi. Neural network and jacobian method for solving the inverse statics of a cable-driven soft arm with nonconstant curvature. *IEEE Transactions on Robotics*, 31(4):823–834, Aug 2015.

[26] M. Giorelli, F. Renda, G. Ferri, and C. Laschi. A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5033–5039, 2013.

[27] A. Handa, M. Bloesch, V. Pătrăucean, S. Stent, J. McCormac, and A. Davison. gvnn: Neural network library for geometric computer vision. In *Proc. European Conference on Computer Vision Workshop on Geometry Meets Deep Learning*, pages 67–82, 2016.

[28] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[29] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[30] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[31] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard. Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *The International Journal of Robotics Research*, 31(12):1445–1464, 2012.

[32] T. A. Huang and M. Kaess. Towards acoustic structure from motion for imaging sonar. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 758–765, 2015.

[33] T. A. Huang and M. Kaess. Incremental data association for acoustic structure from motion. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1334–1341, 2016.

[34] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[35] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[36] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Proc. Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[37] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proc. IEEE 12th International Conference on Computer Vision*, pages 2146–2153, 2009.

[38] Y. Ji, S. Kwak, A. Yamashita, and H. Asama. Acoustic camera-based 3D measurement of underwater objects through automated extraction and association of feature points. In *Proc. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 224–230, 2016.

[39] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv e-prints*, arXiv:1408.5093, 2014.

[40] H. Jiang, Z. Wang, X. Liu, X. Chen, Y. Jin, X. You, and X. Chen. A two-level approach for solving the inverse kinematics of an extensible soft arm considering

viscoelastic behavior. In *Proc. IEEE International Conference on Robotics and Automation*, pages 6127–6133, 2017.

[41] H. Johannsson, M. Kaess, B. Englot, F. Hover, and J. Leonard. Imaging sonar-aided navigation for autonomous underwater harbor surveillance. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4396–4403, 2010.

[42] M. Johnson-Roberson, O. Pizarro, S. B. Williams, and I. Mahon. Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys. *Journal of Field Robotics*, 27(1):21–51, 2010.

[43] J. W. Kaeli, J. J. Leonard, and H. Singh. Visual summaries for low-bandwidth semantic mapping with autonomous underwater vehicles. In *Proc. IEEE/OES Conference on Autonomous Underwater Vehicles*, Oxford, Mississippi, 2014.

[44] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[45] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-DOF camera relocalization. In *Proc. IEEE International Conference on Computer Vision*, pages 2938–2946, 2015.

[46] W. Kier and K. Smith. Tongues, tentacles and trunks: the biomechanics of movement in muscular-hydrostats. *Zoological Journal of the Linnean Society*, 83(4):307–324, 1985.

[47] J. Kim, H. Cho, J. Pyo, B. Kim, and S.-C. Yu. The convolution neural network based agent vehicle detection using forward-looking sonar image. In *Proc. IEEE/MTS OCEANS, Monterey, California*, 2016.

[48] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv e-prints*, arXiv:1412.6980, 2014.

[49] K. Konda and R. Memisevic. Learning visual odometry with a convolutional network. In *Proc. International Conference on Computer Vision Theory and Applications*, Berlin, Germany, 2015.

[50] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, University of Toronto, Department of Computer Science, 2009.

[51] A. Krizhevsky. Convolutional deep belief networks on cifar-10. `https://www.cs.toronto.edu/~kriz/conv-cifar10-aug2010.pdf`, 2010. Unpublished manuscript.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Advances in Neural Information Processing Systems*, pages 1097–1105. 2012.

[53] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[54] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 97–104, June 2004.

[55] E.-H. Lee and S. Lee. Development of underwater terrain's depth map representation method based on occupancy grids with 3D point cloud from polar sonar sensor system. In *Proc. IEEE International Conference on Ubiquitous Robots and Ambient Intelligence*, pages 497–500, 2016.

[56] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[57] B. Li, T. Zhang, and T. Xia. Vehicle detection from 3D LIDAR using fully convolutional network. In *Proc. Robotics: Science and Systems Conference*, Ann Arbor, Michigan, 2016.

[58] H. Lin, G. G. Leisk, and B. Trimmer. GoQBot: a caterpillar-inspired soft-bodied rolling robot. *Bioinspiration & Biomimetics*, 6(2):026007, Apr 2011.

[59] F. Lu. *Shape Registration using Optimization for Mobile Robot Navigation*. PhD thesis, University of Toronto, 1995.

[60] H. Q. Luong, M. Vlaminck, W. Goeman, and W. Philips. Consistent icp for the registration of sparse and inhomogeneous point clouds. In *Proc. IEEE International Conference on Communications and Electronics*, pages 262–267, 2016.

[61] J. Makhoul. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):27–34, 1980.

[62] M. Manti, T. Hassan, G. Passetti, N. D'Elia, C. Laschi, and M. Cianchetti. A bioinspired soft robotic gripper for adaptable and effective grasping. *Soft Robotics*, 2(3):107–116, 2015.

[63] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Proc. International Conference on Artificial Neural Networks*, pages 52–59, 2011.

[64] D. Maturana and S. Scherer. 3d convolutional neural networks for landing zone detection from lidar. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3471–3478, 2015.

[65] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[66] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, 2008.

[67] A. Nicolai and G. A. Hollinger. Denoising autoencoders for laser-based scan registration. *IEEE Robotics and Automation Letters*, 3(4):4391–4398, Oct 2018.

[68] A. Nicolai, G. Olson, Y. Mengüç, and G. A. Hollinger. Learning to control re-configurable staged soft arms. *IEEE Robotics and Automation Letters*, 2019. In Review.

[69] A. Nicolai, R. Skeele, C. Eriksen, and G. A. Hollinger. Deep learning for laser based odometry estimation. In *Proc. Robotics: Science and Systems Conference Workshop on Limits and Potentials of Deep Learning in Robotics*, Ann Arbor, Michigan, 2016.

[70] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–659, 2004.

[71] G. Olson, S. Chow, A. Nicolai, C. Branyan, G. Hollinger, and Y. Mengüç. A generalizable equilibrium model for bending soft arms with longitudinal actuators. *The International Journal of Robotics Research*, 2019. Accepted.

[72] P. Ondruska, J. Dequaire, D. Z. Wang, and I. Posner. End-to-end tracking and semantic segmentation using recurrent neural networks. In *Proc. Robotics: Science and Systems Conference Workshop on Limits and Potentials of Deep Learning in Robotics*, Ann Arbor, Michigan, 2016.

[73] P. Ondruska and I. Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *Proc. AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, 2016.

[74] P. Polygerinos, S. Lyne, Z. Wang, L. F. Nicolini, B. Mosadegh, G. M. Whitesides, and C. J. Walsh. Towards a soft pneumatic glove for hand rehabilitation. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1512–1517, 2013.

[75] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3223–3230, May 2015.

[76] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *Proc. Advances in Neural Information Processing Systems*, page 1137–1144, 2007.

[77] R. Reinhart, Z. Shareef, and J. Steil. Hybrid analytical and data-driven modeling for feed-forward robot control. *Sensors*, 17(2):311, 2017.

[78] F. Renda, M. Cianchetti, M. Giorelli, A. Arienti, and C. Laschi. A 3d steady-state model of a tendon-driven continuum soft manipulator inspired by the octopus arm. *Bioinspiration & Biomimetics*, 7(2):025006, May 2012.

[79] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi. Dynamic model of a multibending soft robot arm driven by cables. *IEEE Transactions on Robotics*, 30(5):1109–1122, Oct 2014.

[80] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, May 2008.

[81] C. Schenck and D. Fox. Detection and tracking of liquids with fully convolutional networks. In *Proc. Robotics: Science and Systems Conference Workshop on Limits and Potentials of Deep Learning in Robotics*, Ann Arbor, Michigan, 2016.

[82] A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Proc. Robotics: Science and Systems Conference*, Seattle, Washington, 2009.

[83] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arxiv e-prints*, arxiv:1409.1556, 2014.

[84] G. Singh, C. Xiao, G. Krishnan, and E. Hsiao-Wecksler. Design and analysis of soft pneumatic sleeve for arm orthosis. In *Proc. ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Charlotte, North Carolina, 2016.

[85] N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In *Learning Theory*, pages 545–560. Springer, 2005.

[86] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[87] G. Tesauro. Practical issues in temporal difference learning. In *Reinforcement Learning*, pages 33–53. Springer, 1992.

[88] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, arxiv:1605.02688, May 2016.

[89] T. Thuruthel, E. Falotico, M. Cianchetti, and Cecilia C. Laschi. Learning global inverse kinematics solutions for a continuum robot. In *Proc. CISM/IFToMM Symposium on Robot Design, Dynamics and Control*, pages 47–54, 2016.

[90] T. Thuruthel, E. Falotico, M. Cianchetti, F. Renda, and C. Laschi. Learning global inverse statics solution for a redundant soft robot. In *Proc. International Conference on Informatics in Control, Automation and Robotics*, pages 303–310, 2016.

[91] T. Thuruthel, E. Falotico, M. Manti, A. Pratesi, M. Cianchetti, and C. Laschi. Learning closed loop kinematic controllers for continuum manipulators in unstructured environments. *Soft Robotics*, 4(3):285–296, 2017.

[92] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, pages 298–375. Springer Verlag, 2000.

[93] T. Umedachi, V. Vikas, and B. Trimmer. Softworms: the design and control of non-pneumatic, 3d-printed, deformable robots. *Bioinspiration & Biomimetics*, 11(2):025001, Mar 2016.

[94] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.

[95] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *arXiv e-prints*, arxiv:1411.455, Nov 2014.

[96] R. J. Webster and B. A. Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.

[97] G. Weiss and E. Puttkamer. A map based on laserscans without geometric interpretation. *Intelligent Autonomous Systems*, pages 403–407, 1995.

[98] T. Weyand, I. Kostrikov, and J. Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55, 2016.

[99] D. P. Williams and S. Dugelay. Multi-view sas image classification using deep learning. In *Proc. IEEE/MTS OCEANS, Monterey, California*, 2016.

[100] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *Proc. of the International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

[101] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv e-prints*, arXiv:1212.5701, 2012.

[102] J. Zhang and S. Singh. LOAM: Lidar odometry and mapping in real-time. In *Proc. Robotics: Science and Systems Conference*, Berkeley, California, 2014.

[103] J. Zhang and S. Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416, 2016.

[104] H. Zhao, K. O'Brien, S. Li, and R. Shepherd. Optoelectronically innervated soft prosthetic hand via stretchable optical waveguides. *Science Robotics*, 1(1), 2016.

[105] R. Zlot and M. Bosse. Efficient large-scale 3D mobile mapping and surface reconstruction of an underground mine. In *Field and Service Robotics*, pages 479–493. Springer, Berlin, Heidelberg, 2014.