A MULTIMODAL EXECUTION MONITOR FOR ASSISTIVE ROBOTS

A Dissertation
Presented to
The Academic Faculty

By

Daehyung Park

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology

May 2018

A MULTIMODAL EXECUTION MONITOR FOR ASSISTIVE ROBOTS

Develop success from failures.

Discouragement and failure are two of the surest stepping stones to success.

Dale Carnegie

To my wife and son

ACKNOWLEDGEMENTS

After an intensive period of five years and half, I finally write this note of thanks to finish my dissertation. I would like to reflect on the people who have supported and helped me so much throughout this period.

I must first thank my family, especially my wife and parents. Without their love and tireless support, this thesis would never have been possible.

The accomplishment of this thesis benefits of the help and direction from my dear advisor—Dr. Charles C. Kemp. Charlie has enthusiastically provided me insightful ideas for developing and presenting the developed system in this thesis. I greatly appreciate Charlie's advice on everything from planning work to writing papers.

Besides my advisor, I would also like to acknowledge insightful guidance and encouragement from the rest of my thesis committee: Byron Boots, Sonia Chernova, James Rehg, and Randy Trumbower. Their comments and questions helped me to widen my research and perspective of thought.

Furthermore, I would like to show my grateful feeling to the fellow members and alumni of Healthcare Robotics Lab. A salute each, in chronological order, to Marc D. Killpack, Kevin Chow, Ashwin A. Shenoi, Yash Chitalia, Phillip M. Grice, Tapomayukh Bhattacharjee, Ariel Kapusta, Zackory Erickson, and Henry M. Clever. I am also very grateful to Henry and Jane Evans for our collaboration while collecting data and evaluating the developed system.

A warm thanks to all Master and Undergraduate students I work with. I am especially grateful to Hokeun Kim and Yuuna Hoshi for helping me develop the robot-assisted feeding system and collecting data for the monitoring system, Chansu Kim for designing tools for the feeding system, You Keun Kim for assisting with the initial condition selection work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Assistive robots have the potential to serve as caregivers, providing assistance with activities of daily living (ADLs) and instrumental activities of daily living (IADLs) to people with disabilities. Yet for many people with disabilities, including the people with upper limb disabilities or quadriplegia, such tasks prove challenging without assistance from a human caregiver. Monitoring when something has gone wrong could help assistive robots operate more safely and effectively around people. However, the complexity of interacting with people and objects in human environments can make challenges in monitoring operations. By monitoring multimodal sensory signals, an execution monitor could perform a variety of roles, such as detecting success, determining when to switch behaviors, and otherwise exhibiting more common sense.

The purpose of this dissertation is to introduce a multimodal execution monitor to improve safety and success of assistive manipulation services. To accomplish this goal, we make three main contributions. First, we introduce a data-driven anomaly detector, a part of the monitor, that reports anomalous task executions from multimodal sensory signals online. The system's anomaly detector models multimodal sensory signals from typical non-anomalous task executions using classic or state-of-the-art machine-learning methods: a hidden Markov model (HMM) and a long-short-term-memory based variational autoencoder (LSTM-VAE). It then determines an anomaly that is biased from the model using a detection threshold that varies based on the state of the task execution.

Second, we introduce a data-driven anomaly classifier that recognizes the type and cause of common anomalies through an artificial neural network after fusing multimodal features. Our method recognizes representative anomalies that are more likely to occur during the direct task at hand. For robot-assisted feeding, we identified 12 anomalies through fault tree analysis which is a deductive analysis approach for resolving system hazards into their causes.

Lastly, as the main testbed of the monitoring system, we introduce a proof-of-concept of a robot-assisted feeding system for people with disabilities. We endow a general-purpose mobile manipulator, a PR2 robot, with the ability to provide safe, easy-to-use feeding assistance. The system enables diverse care receivers to overcome their physical limitations by introducing an autonomous feeding framework in which the PR2 delivers a selected food inside the receiver's mouth. For the framework, we develop vision-based food and mouth detection approaches as well as a web-based user interface. We also use this feeding system for testing our execution monitoring system.

We evaluate the monitoring system with haptic, visual, auditory, and kinematic sensing during household tasks and human-robot interactive tasks including feeding assistance. Our detection methods performed better than other baseline methods from the literature, yielding higher area under receiver operating characteristic curves (AUC) and shorter detection delays. Our classification method also successfully identified the classes of anomalies with higher accuracies. We show multimodality improves the performance of monitoring methods by detecting and classifying a broader range of anomalies. In addition, we evaluate the robot-assisted feeding system with able-bodied participants and people with disabilities in laboratory and real settings. Overall, our research demonstrates the multimodal execution monitoring system helps the assistive manipulation system to provide safe and successful assistance for people with disabilities.

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Overview

Activities of daily living (ADLs), such as feeding and dressing, are important for quality of life [1]. Robotic assistance could help people with disabilities perform ADLs on their own, such as robot-assisted shaving [2], robot-assisted wiping [3, 4], and robot-assisted feeding [5, 6, 7, 8]. Similarly, robotic assistance with instrumental activities of daily living (IADLs) like household chores [9, 10] could be valuable. When operating near humans, it is particularly important that robots safely handle anomalous task executions. A robot can monitor the execution of a task using a separate system that runs in parallel: a type of execution monitoring system [11]. By monitoring multimodal sensory signals, an execution monitor could perform a variety of roles, such as detecting success, determining when to switch behaviors, and otherwise exhibiting more common sense.

In this dissertation, we focus on using an execution monitor for assistive manipulation services. We consider an execution to be anomalous when it is significantly different from typical successful executions, which does not strictly imply that an anomalous execution is a failure. For example, an execution might be anomalous due to high uncertainty about the outcome, but still result in successful completion of the task if allowed to proceed. Due to variability during manipulation in human environments [12], the monitoring of anomalous executions is challenging. An ideal execution monitor should be capable of detecting anomalies online, alerting the robot shortly after the onset of an anomaly, detecting subtle anomalies, ignoring irrelevant task variation, and handling multimodal sensory signals. It should be also capable of identifying the type and cause of anomalies to notify those the robot or user. The produced information will be able to provide opportunities to improve the safety and usability of assistive manipulation systems. In pursuit of these goals, we have developed a data-driven monitoring framework. An assistive robot successfully monitored its operations by detecting and recognizing anomalous executions from multimodal sensory signals.

In the remainder of this introduction, we first address what an anomaly is. We also describe why multimodal execution monitoring is necessary and what issues current monitoring systems have. We then formulate a research problem to construct the multimodal execution monitoring system we introduce in this thesis proposal. Finally, we summarize our key ideas for the system.

1.2 Background

1.2.1 Definition of Anomaly

The Oxford English Dictionary defines an anomaly as "something that deviates from what is standard, normal, or expected." As such, an anomaly can be recognized by several other terminologies: a fault, an outlier, or an unforeseen situation [13, 14, 15]. In the assistive manipulation contexts, we use an anomaly as an anomalous execution that is inconsistent with typical successful task executions. Figure 1.1 shows three representative examples of anomalous executions (i.e., a failed execution, a successful but atypical execution, and an execution with uncertainty) under a reaching task. We call the typical successful execution non-anomalous execution. The blue and red curves show a set of non-anomalous and anomalous trajectories, respectively. The dash curve represents the uncertainty such as the lack of confidence or imprecision in observations.

In robotics, anomalies usually occur around robots and their surroundings. Anomalies come from various sources such as programming bugs, sensing errors, control failures, or environmental changes. Occupation Safety and Health Administration (OSHA) classified the source of robotic anomalies into 6 groups: human error, control error, unauthorized access, mechanical failure, environmental source, and improper installation [16]. In human-

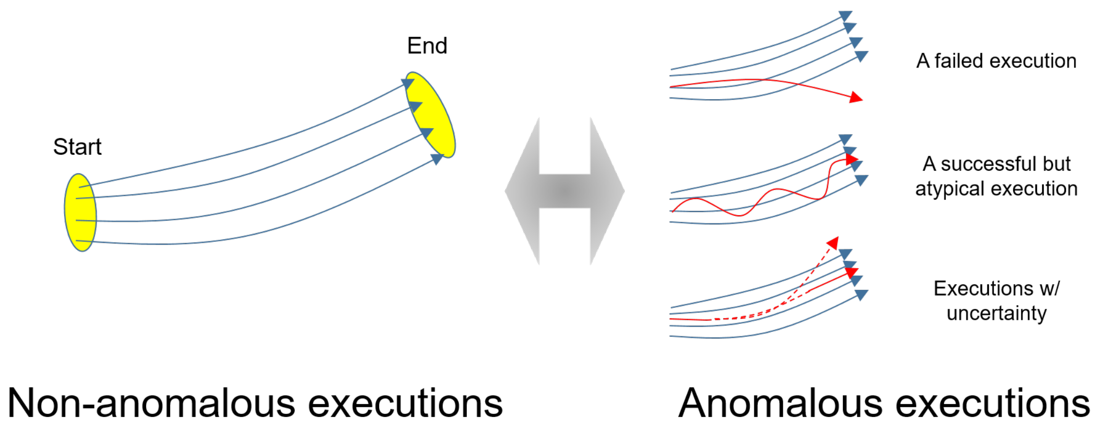


Figure 1.1: Examples of non-anomalous and anomalous executions under a reaching task. **Left**: Blue curves show a pattern of reaching trajectories as typical successful executions (i.e., non-anomalous executions). **Right**: Red curves show examples of anomalous executions.

robot interaction (HRI), researchers have classified the types of anomalies into 3 groups: engineering, human, and environmental conditions [17, 18]. If a robot is able to identify the types, it may be beneficial to reduce or prevent potential hazards. However, anomalies are not easily predictable due to their variabilities; modeling and classification of anomalies is not trivial.

1.2.2 Execution Monitoring

Monitoring of anomalous execution is a form of execution monitoring (an execution monitor). There are a number of definitions of execution monitoring in the literature:

- "An agent's process of identifying discrepancies between observations of the actual world and the prediction and expectations derived from its representation of the world, classifying discrepancies, and recovering from them" [19].
- "Execution monitoring is a continuous real-time task of determining the condition of a physical system, by recording information, recognizing anomalies, and indicating [identifying] them in the behavior" [11].

The former definition assumes the monitor uses a predictive model. The latter definition restricts the monitoring target to the system's behavior, though the status of a task-relevant object or a subject can be an important indicator to determine an anomalous execution. Thus, we define execution monitoring as:

Definition 1. Execution Monitoring A real-time process that determines the status of task executions, by detecting and recognizing anomalous executions from observed task-relevant information.

To observe the system itself, the target, or the environment, a monitoring system retrieves task-relevant information from multiple sensory inputs. We call this framework multimodal execution monitoring.

1.3 Opportunities of Execution Monitoring

To prevent anomalies, industrial robots usually operate in inaccessible environments surrounded by barriers, such as physical fences, to separate people from the robots' workspace [20], and execute repetitive tasks with few uncertainties. However, the growth of service or collaborative robots exposes robots to anomalies frequently. A lack of detection and recovery systems may lower the usage of robots due to potential failure cost.

As a solution, researchers have been widely studying execution monitoring system in a variety of domains: programming [21], robotics [11], energy management [22], and business management [23]. In robotics, a number of researchers have studied this topic in the areas of control, planning, and high-level programming [24, 25, 26, 27]. An ideal execution monitor should

- be capable of detecting anomalies online, alerting the robot shortly after the onset of an anomaly, working for long-duration behaviors, detecting subtle anomalies, ignoring irrelevant task variation, and handling multimodal sensory signals, and

- be capable of identifying one or more types of anomalies, determining a primary type, ignoring subsequent types, and providing interpretable information to both a user and system to correct an anomalous execution.

The monitor will enable robots to achieve effectiveness, efficiency, and satisfaction in assistive manipulation contexts. In pursuit of these goals, we present a data-driven monitoring framework using multimodality.

1.4 Problem Description

1.4.1 Multimodal Execution Monitoring

We define the problem of multimodal execution monitoring as the detection of anomalous executions and the classification of the detected anomalies from multiple sensory signals associated with the executed behavior. An execution monitor should detect when the current sensory signals differ significantly from past sensory signals associated with task successes, which is analogous to the problem of finding unexpected patterns in data, called anomaly, failure, or outlier detections in various domains [14, 15]. The monitor should also identify the type, size, or location of anomalies so that they can be used to prevent potential failure, minimize damage (or injury), or correct the executions in assistive manipulation contexts.

1.4.2 Challenges and Issues

Although suitable for a narrower set of situations, stereotyped task-specific behaviors tend to have lower variability in their operation than more general methods, which can reduce the variation of the associated multimodal signals, thereby lowering data requirements for data-driven methods and simplifying anomaly detection and classification [28, 29]. As a result of the complexities of real-world manipulation, anomaly detection and classification are challenging. We address several factors that make the monitoring difficult in assitive manipulation contexts:

- A wide variety of anomalies exist but most anomalies occur sporadically. The collection of anomalies is a non-trivial problem in assistive manipulation. No single sensor can observe all anomalies.
- The variability of tasks, users, and environments makes it difficult for the monitor to distinguish the non-anomalous and anomalous executions.
- Subtle anomalies are not easily distinguishable from the noise or variation of non-anomalous executions. However, subtle anomalies may result in hazards during the assistance.
- Fast detection is an important factor for the safety of users but it may increase false alarms.
- Feature selection (or engineering) is a frequent problem in anomaly detection and classification. The determination of possibly relevant features is non-trivial given heterogeneous sensor data.
- A system with cheap or noisy sensors is easily prone to low confidence or imprecision in decision.
- An anomaly usually results in a series and mixture of problems so it is not easily identifiable.

In an attempt to address these concerns, we present a multimodal execution monitoring system.

1.5 Contributions

The contributions of this dissertation are the development of an online execution monitoring system using multimodalities for an assistive robot that performs IADLs or ADLs. We introduce individual monitoring frameworks and their applications.

1.5.1 Multimodal Anomaly Detection

We introduce anomaly detectors, as a part of the monitoring system, using multimodality. We are interested in investigating the effectiveness of multimodalities, especially in detecting anomalies during assistive tasks. We are also interested in data-driven approaches that jointly models multimodal time-series signals from typical non-anomalous tasks and detect an anomaly that is largely biased from the model. For it, we first introduce a detector that uses a classic machine-learning method, multivariate Gaussian hidden Markov models (HMMs) with a likelihood-based classifier. We also introduce another detector that consists of the state-of-the-art deep-learning methods, long-short term memory (LSTM) networks and a variational autoencoder (VAE), that do not require the effort of significant feature engineering. To increase detection sensitivity and lower false alarms, we change decision boundaries over the progress of task executions. We show our methods outperform baseline methods in literature.

1.5.2 Multimodal Anomaly Classification

For anomaly classification, we introduce a multimodal identification method for known anomalies using a data-driven classifier. In particular, we show how to fuse multimodal signals and image data to train the classifier. After the detection of anomalies, the classifier estimates the most probable type of anomalies among the known classes. In this dissertation, we pick identifiable types of anomalies using fault tree analysis (FTA). We demonstrate that our classification method successfully identifies the types of anomalies from 12 identifiable anomalies in the assistive feeding task.

To evaluate our methods, we have been collecting multimodal signals such as haptic, auditory, kinematic, and vision signals while a PR2 robot performed pushing and feeding tasks, which are representative examples of instrumental activities of daily living and activities of daily living, respectively [30, 31, 8]. We evaluate our method with a PR2 robot, which repeatedly closed the door of a microwave oven or the lid of a tool case. We also run a study in which a PR2 robot fed yogurt to able-bodied participants, scooping yogurt from a bowl and bringing it to their mouths. We will compare our multimodal monitoring to other baseline monitoring methods and show it obtains a higher area under curve (AUC) from receiver operating characteristic (ROC) curves and shorter detection delays with simulated and experimental anomalous data. We will also show that multimodalities substantially improved detection and classification performance.

1.5.3 Application: A Robot-Assisted Feeding System

As a test bed for the execution monitoring system, we present a proof-of-concept robotic system for assistive feeding that consists of a Willow Garage PR2, a high-level web-based graphical user interface, and specialized autonomous behaviors for scooping/stabbing and feeding foods. Our feeding system contributes to the adoption of the general-purpose mobile manipulator as an assistive robot. Unlike other commercial or research platforms, our system provides *active feeding* assistance for diverse people with disabilities, where *active feeding* is that an caregiver or a robot autonomously deliver a user-selected food inside the user's mouth.

As a step towards use by people with disabilities, we evaluate our system with 9 able-bodied participants as well as the author. Also, Henry Evans, a person with severe quadriplegia, operates the system remotely to feed an able-bodied person or directly to feed himself at home as a pilot study. Finally, we evaluate the system with people with disabilities. We will compare the two groups of evaluation results and show the system is convenient and easy-to-use. We will also show the system provides safe and comfortable food delivery.

1.6 Organization of Thesis

We organize the remainder of this thesis as follows. We first present the related work relevant to our multimodal execution monitoring, especially including anomaly detection and classification. We also review assistance systems relevant to the robot-assisted feeding as a main testbed.

Then, Chapter 3 presents our approaches to detect anomalies, which emphasizes the benefit of using multimodality in the assistive manipulation contexts. We assume that the stereo-type assistive manipulations convey a typical pattern of signals. We develop two approaches that model multimodal time-series signals using HMMs and VAE with LSTMs (LSTM-VAE), respectively. The approaches detect an anomaly when signals are biased from the trained model. We evaluate the approaches in object pushing tasks and robot-assisted scooping-and-feeding tasks while collecting 5 modalities. We show the multimodality helps to increase the performance of anomaly detection with low false alarms. We also show our detectors performs better than baseline methods in literature.

Chapter 4 presents an anomaly classifier that enables a robot to identify anomalies. We assume that more modalities and their derived features help to precisely determine the type and cause of anomalies during assistive manipulation, particularly the robot-assisted feeding. We develop a classification network that extracts and combines temporal and convolutional features using multi-layer perceptrons. We evaluate the classifier in the robot-assisted feeding task with able-bodied participants. We show the multimodality helps to successfully classify a set of common occurring anomalies during the feeding task. In addition, we also show an in-home evaluation with a person with quadriplegia.

Chapter 5 presents a proof-of-concept robot-assisted feeding system that is a main testbed of our execution monitor. We describe the detail of individual sub tasks such as scooping, wiping, and feeding. We also present estimation and interaction components that enable the system to automatically and robustly provide the feeding assistance to diverse users. We then present the study design and results of laboratory evaluations with multiple able-bodied participants and in-home evaluation with a person with quadriplegia. We examine the participants' use of the system.

Chapter 6 summarizes this work and its results. We then briefly discuss the potential impact. In Appendix A, we provide links to code, videos, and data of this thesis.

CHAPTER 2
RELATED WORK

2.1 Multimodal Time-Series Data

2.1.1 Multimodality

Researchers have found that distinct human sensory modalities can be closely coupled [32]. Inspired by this research, Fitzpatrick et al. introduced a crossmodal method that enabled a robot to learn relationships between visual and auditory signals while manipulating objects [33]. Wu and Siegel investigated the use of combining acceleration and sound measurements to detect structural defects in airplane components [34]. Su et al. introduced a multimodal event detection framework for peg-in-hole tasks using a robot's end-effector position and tactile sensors [35]. Wade et al. showed material-recognition framework using a fabric-based skin with force and thermal sensors [36].

2.1.2 Time-Series Data

Researchers have taken two distinct approaches to modeling time-series sensory signals. In the first, a model learns its internal structure from each sensory modality independently. Rodriguez et al. classified the failure of a robotic assembly by extracting a unimodal signal, the magnitude of force output [37]. Ando et al. determined the anomalous behaviors of a mobile robot using k-nearest neighbors, which measures the difference between modalities independently [38]. Pastor et al. used multimodal sensory signals to predict failures while making a robot flip a box using chopsticks. Their method independently modeled each signal using dynamic movement primitives and predicted a failure when five or more signals failed independent z-tests for three consecutive time steps with respect to recorded signals from successful attempts [39]. Chu et al. trained multiple HMMs with respect to each modality and input a set of log-probabilities from the HMMs to a linear SVM for haptic adjective classification [40]. In the second, a model learns multivariate representations from multimodal sensory signals. Clifton et al. represented the multimodal distribution of data using multivariate Gaussian mixture models (GMM) and detected anomalies using extreme value theory [41]. In our previous work [42], we modeled multimodal sensing signals using the multivariate Gaussian emission probabilities of HMMs. This approach, although computationally expensive, is capable of checking anomalies in which two or more signals jointly change.

2.2 Execution Monitoring

Execution monitoring has been well-studied in robotics for detecting and classifying anomalous executions [11, 13]. Bjreland introduced a monitoring system that detects, classifies, and corrects anomalous executions using a predictive model [19]. Pettersson introduced another monitoring system which detects and indicates anomalies in robot behaviors without restricting the detection method to a predictive model [43]. Unlike this previous work, our execution monitor observes the status of a task-relevant object and a person.

2.2.1 State Estimation

Monitoring methods usually estimate the status of a system from observations and determine anomalies with respect to that state. For example, Jain and Kemp estimated a task-centric state (i.e., opening angle) from observed kinematics during a door-opening task and detected blocked or locked doors [29]. Haidu et al. split a trajectory into a number of bins as states and determined anomaly decision boundaries per bin [44]. Likewise, Kappler et al. split multimodal signals, such as force, audio, and kinematics data, into discretized time states and trained a linear support vector machine as a failure classifier per bin [45]. Simanek et al. estimated the state of a mobile robot from multimodal data using an extended Kalman filter and rejected anomalous data using statistical tests [46]. The mentioned methods assumed that a state could only be determined by current observations. Unlike the directly observable states they used, we represent the current progress of a task execution as the distribution over hidden states of an HMM.

2.2.2 Anomaly Detection

Overview Anomaly detection, the main role of execution monitoring, has been widely explored in various domains, including credit-card fraud, cyber intrusion, and novelty detection [47, 14, 48, 49]. A number of robotics researchers have also applied it to robotics applications: mechanical-failure detection [50, 51], sensor-fault detection [52, 53], and environmental anomaly detection [54, 55]. These studies have also been applied to the detection of the anomalous statuses of highly complex systems such as humanoid robot falls [56, 57]. As an extension, similar to our work, researchers are attempting to detect task-dependent failures: blocked door [29] and bin-picking failure [58]. For example, Sukhoy et al. detected manipulation failures in a magnetic card sliding task using multiple torque outputs [59]. Rather than using static decision boundaries, we use dynamic decision boundaries that depend on the currently perceived state of the task to improve the detection performance.

Anomaly detection has been investigated for various assistive devices. For example, Geravand and et al. introduced a fall detector that monitors force-torque data to predict and prevent falling during mobility assistance [60]. Colombo et al. showed environmental anomaly detection (e.g. wet floors, road block, or a change in environment) using a visual modality with a robotic walker, DALi [61]. We also introduced an anomaly detector that checks multimodal sensory signals to monitor robot assistance, such as robot-assisted feeding [42]. We have also successfully tested our new system with a person with disabilities.

Classic Methods Many classic machine learning approaches have also been used: support vector machine (SVM) [37, 62], self-organizing map (SOM) [63], k-nearest neighbors (kNN) [38], etc. To detect anomalies from time-series signals, researchers have also used hidden Markov models [42, 64] or Kalman filters [65].

Our research is closely related to time-series data classification. Fujii et al. determined material defects in hammering tests by continuously matching time-frequency audio images with template patterns [66]. Niekum et al. detected motion change points using the evidence probabilities of articulation models given a segment of time-series data [67]. Similar to our work, before classification, a number of researchers modeled time-series data using dynamics models. Fagogenis et al. modeled time-series data using locally weighted projection regression to check the thruster failures of autonomous underwater vehicles [68]. Researchers also used Markovian modeling [14] and computed likelihoods, state paths, or state distributions to classify anomalies over time [69, 70]. In robotics, Morris and Trivedi used an HMM to model object paths for abnormal behaviors [71]. Mendoza et al. also used an HMM to model multiple signals such as velocity, acceleration, and jerk signals [72]. However, different from our approach, these studies assumed conditional independence between signals even though the signals may have been correlated. We instead use full covariance between signals in HMMs.

Deep Learning Researchers have often fused high-dimensional inputs and reduced their dimension using kernel-based approaches before applying probabilistic or distance-based detections [37, 59, 73]. However, the compressed representations of outliers (i.e., anomalous data) may be inliers in latent space. Instead, we use a reconstruction-based method that recovers inputs from its compressed representation so that it can measure reconstruction error with the anomaly score. An AE is a representative reconstruction approach that is a connected network with an encoder and a decoder [74]. It has also been applied for reconstructing time-series data using a sliding time-window [75]. However, the window method does not represent dependencies between nearby windows and a window may not contain an anomaly.

To model time-series data with its temporal dependencies, we use an LSTM network [76], which is a type of recurrent neural network (RNN). The advantages of LSTM networks over classic approaches such as window approaches or Markov chains are the representation power and the memory to track longer term dependencies. In contrast to the HMMs, LSTM networks are able to use unlimited states. Researchers have used LSTM networks for prediction in these anomaly detection domains: radio anomaly detection [77] and EEG signal anomaly detection [78]. Malhorta et al. introduced an LSTM-based anomaly detector (LSTM-AD) that measures the distribution of prediction errors [79]. However, the method may not predict time-series under unpredictable external changes such as manual control and load on a machine [80]. Alternatively, researchers have introduced RNN- and LSTM-based autoencoders for reconstruction-based anomaly detection [81, 82]. In particular, Malhorta et al. introduced an LSTM-based encoder-and-decoder (EncDec-AD) that estimates reconstruction error [80]. We also use this reconstruction scheme as a baseline method in this dissertation.

Another relevant approach is a variational autoencoder (VAE) [83]. Unlike an AE, a VAE models the underlying probability distribution of observations using variational inference (VI). Bayer and Osendorfer used VI to learn the underlying distribution of sequences and introduced stochastic recurrent networks [84]. Soelch et al. used their work to detect robot anomalies by predicting unimodal signals [85]. Bowman and Vilnis introduced an RNN-based VAE for language generation [86]. Our work also uses variational inference, but we estimate the expected distribution of input signals and a corresponding state in latent space for state-based thresholding and anomaly detection at each time step.

2.2.3 Anomaly Classification

Anomaly classification is also known as fault isolation or diagnosis [87, 88] and is part of the fault detection and isolation defined by IFAC SAFEPROCESS committee in 1993. As we discuss below, the classification has been used to determine the source of anomalies while running manipulators or mobile robots [89]. Based on Pettersson's classification [11], we can classify relevant work into three groups: causal analysis, expert systems, and data-driven classification. Causal analysis finds the cause of an anomaly based on the relationship between the fault and the cause (e.g., a signed directed graph [90]). Expert systems are widely applied in industry to isolate the cause of an anomaly using "IF THEN" rules or Fuzzy logic [91]. These two approaches often make use of extensive, detailed programming by domain experts. On the other hand, a data-driven approach is feasible if anomaly data are available. Several researchers have applied neural network-based classifiers for anomaly classification to robotic manipulation tasks, such as [92, 93]. Yamazaki et al. performed a database search for the closest anomaly type given a detected failure in a robot-assisted dressing task. [94]. We also use a data-driven classifier to fuse multimodal sensory data and classify anomalies of known classes.

Multimodal fusion and classification are also a closely related area. Ngiam et al. introduced a multimodal fusion method that generates a shared representation between modalities using a single network [95]. They discussed three different levels of fusions: early, intermediate, and late fusion. In this dissertation, we use late fusion with an MLP. Sung et al. showed a multimodal classification method that finds a desired trajectory from a feature space where one or more modalities are separately embedded [96]. We also embed multimodalities into a space after concatenating modalities.

2.2.4 Thresholding

Researchers have determined a fault when a raw sensory signal exceeds a fixed threshold [97]. However, due to the range of the signals in various operations, a threshold had to be large enough to minimize frequent false alarms. Narrowing down the range of operations, Serdio et al. introduced a time-varying threshold, allowing a tight decision boundary around low variance area for rolling mills [98]. However, time variations, such as delay or speed variation, easily break down the detection system.

Instead, researchers have modeled time-series signals and determined anomalies or new events when the likelihood of current observations is lower than a fixed threshold [99, 100]. For example, Vaswani et al. detected abnormal activities when the expected negative log-likelihood was lower than a fixed threshold [101]. Lello et al. detected anomalous force signals using a fixed threshold [102], and Ocak et al. classified anomalies when either likelihood exceeded a fixed threshold or a change in the likelihood between time steps exceeded another fixed threshold [103]. However, these thresholding methods are not applicable if the likelihood does not drop significantly or quickly. Setting a global threshold may reject a number of non-anomalous executions, though it may also fail to detect all anomalies [47].

To address these concerns, Rodriguez et al. set cutoff probability thresholds over discretized time slices [58]. Yeung and Ding used a varying likelihood threshold based on the probabilistic distribution of discrete observations [104]. The use of a time-varying likelihood threshold is similar to our work, but we do not directly use observations. Instead, we vary the threshold with respect to the hidden-state distribution, execution progress, estimated by an HMM.

2.3 Assistive Robots

Assistive robots are a type of devices that can provide physical, mental, or social assistance to people with disabilities or seniors [105, 106]. Assistive Robots have the potential to provide various ADLs such as scratching, brushing, dressing, and feeding tasks [107, 31, 108]. In this section, we review assistive manipulators for ADLs. We then go over feeding devices including assistive feeding arms.

2.3.1 Assistive Manipulators

Researchers have introduced a variety of assistive manipulators—such as 7-DoF arms mounted on a wheelchair or desk—to provide general assistance near the human [109, 110, 111, 112, 113]. We categorize the types of manipulators in terms of mobility: fixed- and mobile-base manipulators.

Fixed-base Robots Fixed-base assistive robots are often placed nearby a user or a targeted workspace. In early stage of assistive robotics, researchers have mounted assistive manipulators on desktop for feeding, cosmetic, hygiene. The professional vocational assistive robot (ProVAR) is a representative desktop manipulator placed in an office workspace [110]. Handy-1 is another adjustable table-mounted manipulator for ADLs such as feeding, drinking, and washing applications [31]. The mounted robots are designed to perform various ADLs using a general-purpose manipulator. However, their limited workspace restricts the range of available activities. Alternatively, researchers have introduced various wheelchair-mounted assistive robots (WMARs). For feeding assistance, Maheu et al. showed that people with disability can feed themselves using a manually controlled JACO arm mounted on a wheel chair [114]. Schroer et al. showed drinking assistance using 7-DoF KUKA arm [115]. For object fetching, Kim et al. also introduced a UCF-MANUS robot, a wheelchair mounted manipulator and an interface [116].

Mobile-base Robots The absence of mobility is an important issue in robotic assistance. Hawkins et al. found that a manipulator requires to use multiple base position for assistance [2]. In feeding, the fixed robot base often requires the relocation of robot or user by caregivers in the beginning or during the task. A fixed base restricts the scope of assistive tasks [118]. The robots are restricted to a narrow set of tasks and are unable to leave the immediate vicinity of the human to provide assistance elsewhere. Recent studies have introduced general-purpose mobile manipulators for various assistive robotic tasks, including shaving, picking-and-placing, and guiding tasks [119, 2, 120, 121, 122, 123]. Our feeding system also has a mobile base that has the potential to enhance the quality of feeding assistance.

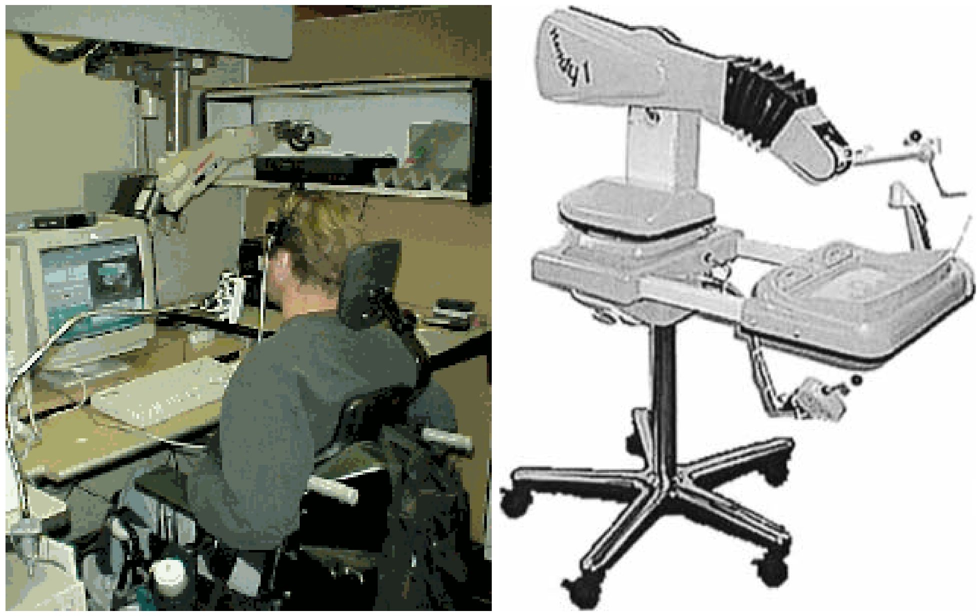


Figure 2.1: **Left**: ProVAR desktop manipulation system described in [110]. **Right**: Handy-1 copied from [31].

2.3.2 Feeding Assistance Devices

Researchers have introduced various assistance devices for feeding from arm supports to feeding robots.

Arm Support Arm support devices enable users to use their upper limb by supporting their weak arm or damping tremor [124, 125]. The devices can be powered or underpowered but require users' manual movements using their upper limb. Thus, depending on individuals with disabilities, the comfortability and efficiency of feeding varies largely.

Feeding Robots The use of feeding robots is an alternative solution for various levels of people with disabilities. A number of commercially available solutions exist: Handy-1 [31], Winsford feeder [126], My Spoon [127], Bestic arm [128], Mealtime partner [129], Meal buddy [130], etc. These robots are designed for a particular purpose (i.e., feeding), often having a desk-mountable fixed base and a low degree-of-freedom arm. A user can command a sequence of scooping-feeding motions via a joystick or a button. The robots

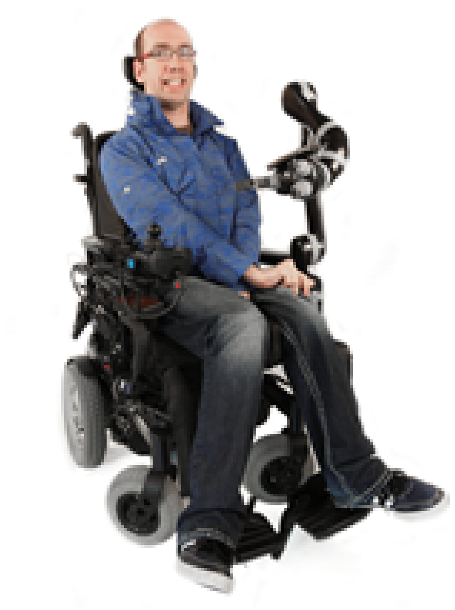
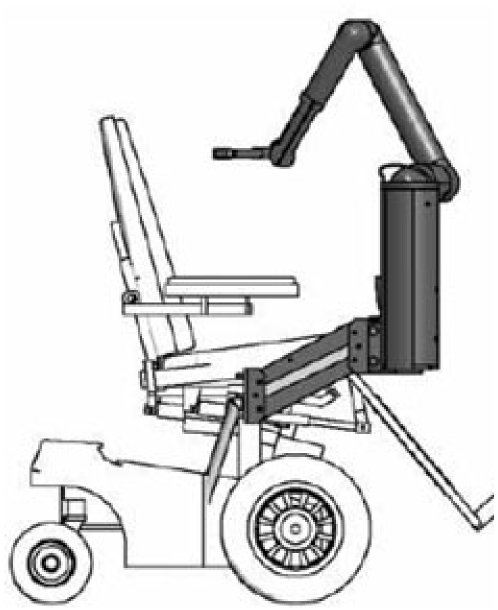



Figure 2.2: **Left**: JACO robotic arm mounted on a powered wheelchair copied from [117]. **Right**: UCF-MANUS copied from [116].

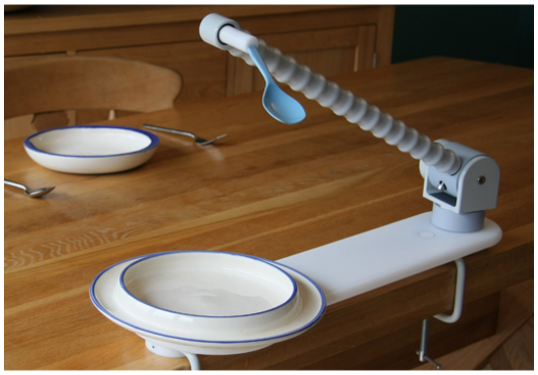


Figure 2.3: **Left**: Manual neater copied from [124]. **Right**: Liftware copied from [125].

perform pre-defined motions where food and mouth are placed in a predefined location. A recently released robot, Obi [131], provides a hand-guided mouth location teaching from caregivers. However, there is no robot which provide teaching or locating for a person with upper limb disability.

Researchers have introduced advanced feeding assistance system with various functionalities [132]. Song and Kim designed a feeding robot with a specialized gripper for Korean food [133]. Yamazaki et al. introduced a 5-DoF meal-assistance robot that provides various food-taking movements [134, 135]. The robot also allows each user to select a desired

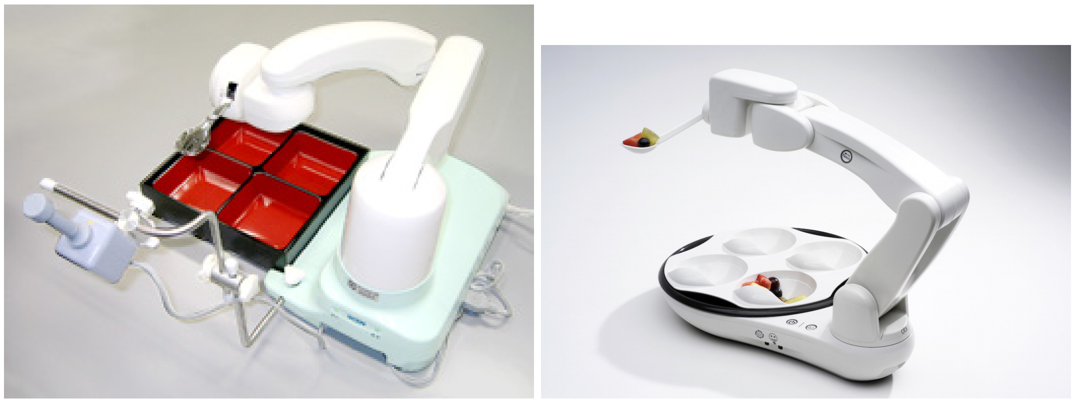


Figure 2.4: **Left**: My Spoon copied from [127]. **Right**: Obi copied from [131].

food-taking location via a graphical user interface (GUI). Recently, Admoni and Srinivasa introduced a gaze-based shared autonomy framework to predict a user's target piece of food and retrieve it [136]. Unlike these manual or semi autonomous systems, Kobayashi et al. introduced an automatic remnant food scooping method using a laser range finder [137]. Similarly, our system also determine the best scooping location using an RGB-D camera and autonomously retrieve it.

In terms of the mouth selection, most robotic systems use passive feeding executions in which the robots convey food into a pre-defined location in front of mouth. Thus, it requires users' upper body movement to take the food. Takahashi and Suzukawa, on the other hand, introduced a feeding system with an interface enabling a user with quadriplegia to manually adjust feeding locations [138]. Similar to our work, Schroer et al. proposed an adaptive drinking assistance robot that finds the user's mouth with a vision system [115]. In addition, Canal et al. adapted feeding movements to users' preferences by incrementally updating movement primitives [139]. Table 2.1 shows a comparison result of features in currently available feeding robots.

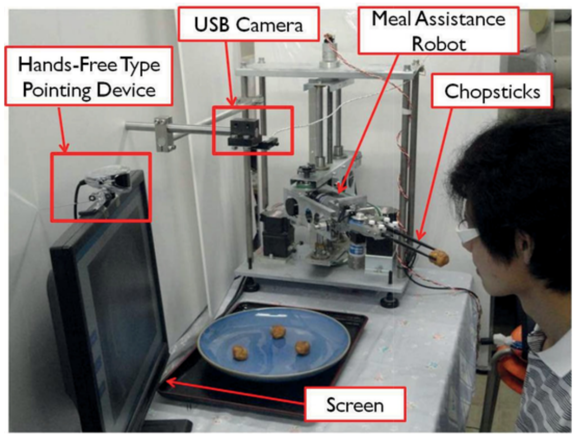
2.3.3 Safety

Assistive feeding robots require to contact with users' mouth that is easily vulnerable to physical contact. Researchers have introduced several safety features to prevent or mini-

Table 2.1: A list of features in robot-assisted feeding systems

	Platform	Interface[a]	Tools[b]	Movement Type		Base	Safety
				scooping	feeding		
Commercial	My Spoon [127]	Joystick	sf	Predefined		Fixed	-
	Bestic arm [128]	Button	s	Predefined		Fixed	-
	Meal Buddy [130]	Joystick	s	Predefined		Fixed	-
	Mealtime [129]	Button	s	Predefined		Fixed	A shatterproof spoon
	Obi [131]	Button	s	Predefined	Hand-guided	Fixed	Collision detection
Academic	Yamazaki and Masuda [134]	GUI(H)	sfc	User-selected	Predefined	Fixed	Force detection
	Song and Kim [133]	Joystick Button	sg	Predefined	Predefined	Fixed	-
	Schroer et al. [115][c]	Brain machine	N/A	N/A	Visually guided	Movable	-
	Kobayashi et al. [137]	Touch sensor	sc	Visually guided	Predefined	Fixed	Spring joint
	Perera et al. [140, 141]	Brain machine	s	Predefined	Predefined	Fixed	-
	Admoni and Srinivasa [136]	Joystick or Gaze	f	User-selected	Predefined	Fixed	-
	Our Work	GUI(H) or Gesture(H)	sf	Predefined / visually guided	Visually guided	Movable	Anomaly detection

[a] H: A head tracker is used as a pointing device, E: An eye tracker is used as pointing device.
[b] s: spoon, f: fork, c: chopstick, g: gripper.
[c] Drinking task only.



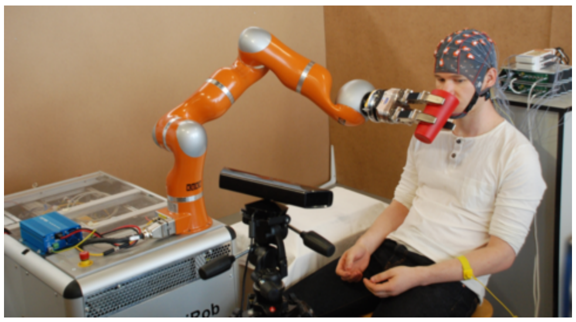


Figure 2.5: **Left**: Yamazaki et al. [135] **Right**: Schroer et al. [115].

mize hazards. A common approach is the reduction of tool stiffness. Researchers attached tools using magnet, spring, or tube [133, 134, 137]. Instead, our system also uses a flexible silicone spoon [8]. In addition to the hardware, researchers applied collision detectors. For example, the Obi feeding robot from Eclipse Automation determines collision using joint sensors [131]. Yamazaki et al. detected unexpected force against to its approaching direction during food taking [134]. However, contact force varies largely depending on users' eating patterns, feeding tools, and foods. The contact should be determined under contexts. Our previous work introduced a multimodal anomaly detector that determines an anomaly by modeling non-anomalous feeding patterns [42, 142]. The detector enables the system stop and return the tool into a safe location.

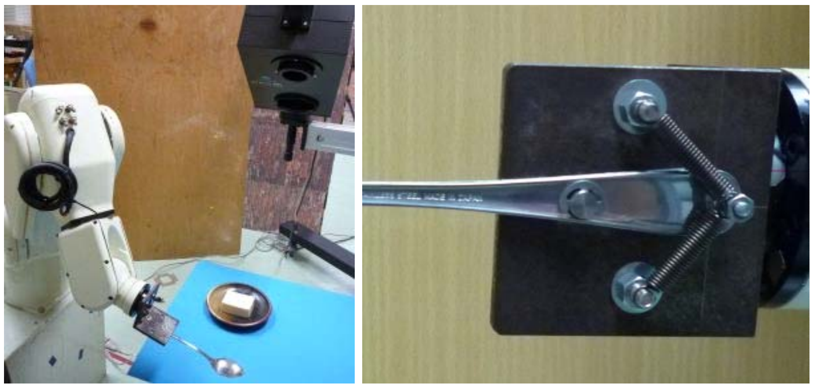


Figure 2.6: A meal support system and its spring-attached spoon copied from [137]

CHAPTER 3
MULTIMODAL ANOMALY DETECTION

3.1 Overview

This chapter focuses on providing evidence for the value of multimodal anomaly detection and the use of a detection threshold that varies based on the progress of task execution.

Online detection of anomalous execution can be valuable for robot manipulation, enabling robots to operate more safely, determine when a behavior is inappropriate, and otherwise exhibit more common sense. By using multiple complementary sensory modalities, robots could potentially detect a wider variety of anomalies, such as anomalous contact or a loud utterance by a human. However, task variability and the potential for false positives make online anomaly detection challenging, especially for long-duration manipulation behaviors. An ideal anomaly detector should be capable of detecting anomalies online, alerting the robot shortly after the onset of an anomaly, working for long-duration behaviors, detecting subtle anomalies, ignoring irrelevant task variation, and handling multimodal sensory signals. In pursuit of these goal, we introduce data-driven anomaly detection frameworks that uses multimodalities.

We first introduce an anomaly detector that uses multivariate Gaussian hidden Markov models (HMMs) to jointly model multimodal time-series signals. We train an HMM on non-anomalous executions. In testing, we use a statistical approach that determines the difference level of the current log-likelihood from a set of estimated log-likelihoods from non-anomalous executions. To handle task variability, detect anomalies quickly, and reduce the number of false alarms, we also introduce two threshold estimation methods based on clustering and regression. The clustering-based method groups likelihood and *execution progress* pairs into a number of clusters and retrieves a statistically expected likelihood with respect to the current *execution progress*. We call this method HMM-D. We also introduce a regression-based method, which maps *execution progress* to likelihood using Gaussian process regression. It then regresses the expected likelihood with respect to the current *execution progress*. We call this method HMM-GP. Both methods perform probabilistic decision making that computes the confidence interval of the expected likelihood and uses the interval as the decision boundary (i.e., the detection threshold).

To evaluate our methods, we collected multimodal signals such as haptic, auditory, kinematic, and vision signals while a PR2 robot performed pushing and feeding tasks for IADLs and ADLs, respectively [30, 31]. We evaluated our method with a PR2 robot, which repeatedly closed the door of a microwave oven or the lid of a toolbox. We also ran a study in which a PR2 robot fed yogurt to able-bodied participants, scooping yogurt from a bowl and bringing it to their mouths (see Figure 3.1). In evaluations, we compared several performance scores with simulated and experimental anomalous data. Our multimodal monitoring outperformed other baseline monitoring methods, obtaining a higher area under curve (AUC) from receiver operating characteristic (ROC) curves and shorter detection delays. We also show that multimodalities substantially improved detection performance and detected a broad range of anomalies.

In addition to the HMM-based detector, we introduce a long short-term memory-based variational autoencoder (LSTM-VAE) for multimodal anomaly detection. We call LSTM-VAE based anomaly detector. Although this method requires a large number of data, this does not requires the effort of significant feature engineering and domain expertise like the previous method. For encoding, an LSTM-VAE projects multimodal observations and their temporal dependencies at each time step into a latent space using serially connected LSTM and VAE layers. For decoding, it estimates the expected distribution of the multimodal inputs from the latent space representation. We train it under a denoising autoencoding criterion [143] to prevent learning an identity function and improve representation capability. Our LSTM-VAE-based detector detects an anomaly when the log-likelihood of current

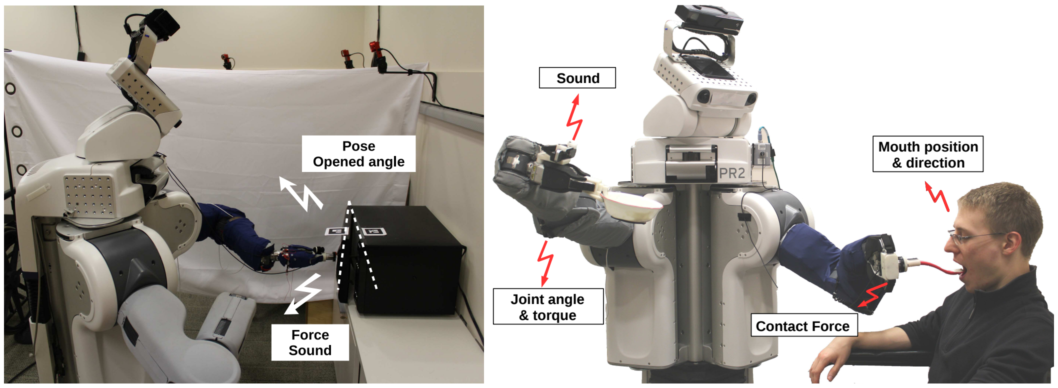


Figure 3.1: Multimodal monitoring in a robotic door-closing and feeding tasks. A PR2 robot determines anomalous execution of the task collecting haptic, auditory, kinematic, and visual sensory signals.

observation given the expected distribution is lower than a threshold. We also introduce a state-based threshold to increase detection sensitivity and lower the false alarms.

We evaluated the LSTM-VAE with robot-assisted feeding data that we collected from 24 able-bodied participants with 1,555 feeding executions. The proposed detector is beneficial in that we could directly use high-dimensional multimodal sensory signals without significant effort for feature engineering. It was able to catch an anomaly online. In particular, it was able to set tight or loose decision boundaries depending on the variations of multimodal signals using the state-based threshold. Our method had higher area under receiver operating characteristic (ROC) curves than other baseline methods from the literature. In our evaluation, the area under the curve (AUC) was 0.044 higher than that of our previous algorithm, HMM-GP given the same data. Our new method also had a 0.064 higher AUC when we used 17-dimensional sensory signals from visual, haptic, kinematic, and auditory modalities instead of 4-dimensional hand-engineered features.

3.2 Multimodal Anomaly Detection I: HMMs

We introduce a multimodal anomaly detector and a detection threshold that varies based on the progress of task execution [42, 142]. Using a data-driven approach, we train an anomaly detector that runs in parallel to a manipulation behavior (see Figure 3.2). Simi-

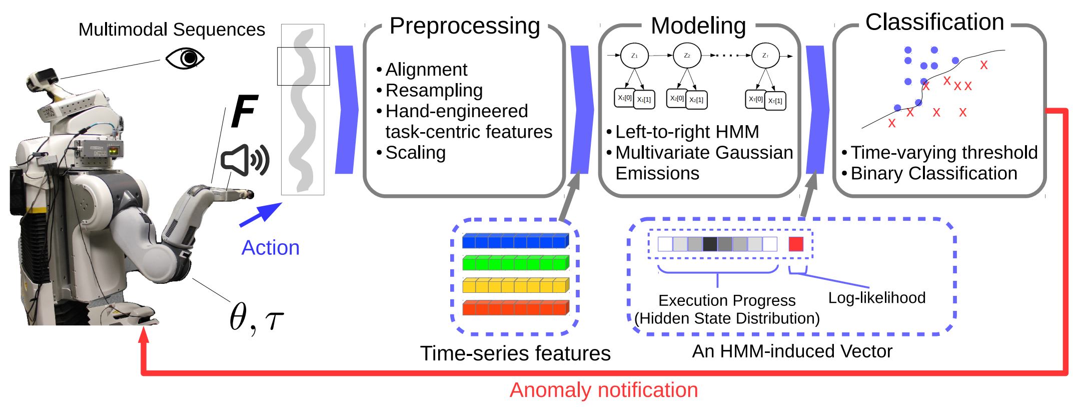
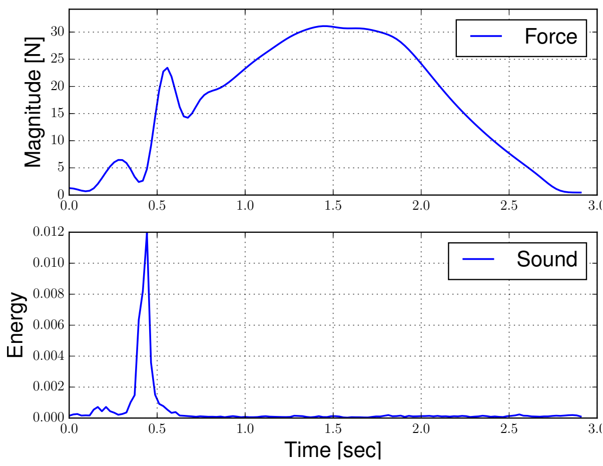


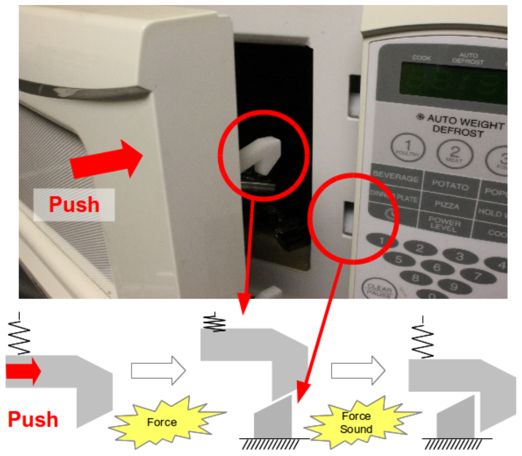
Figure 3.2: The framework of our proposed execution monitoring system for anomaly detection.

lar to classic approaches in literature, our method trains a hidden Markov model (HMM) using multimodal sensory signals recorded during non-anomalous executions [144, 145]. For a particular HMM, all signals come from executions of a specific robot behavior (e.g., pushing or feeding) applied to a specific task (e.g., closing a door or feeding yogurt) performed with specific objects (e.g., a particular microwave oven or a particular person). Our approach could potentially generalize to categories of objects, but for this section we only consider specific objects with which the robot has already had experience. At run time, an HMM provides likelihood estimates, which our system compares to a detection threshold that is based on a probabilistic representation of execution progress. If at any time the log-likelihood is below the current detection threshold, our system detects an anomaly.

In details, we introduce two different types of detection algorithms: clustering- and regression-based detectors. The former detector uses clusters to estimate the mean and standard deviation of the HMM's log-likelihood given the execution progress. We call this method HMM-D. Cluster membership for HMM-D is based on temporal similarity. We also introduce another clustering-based algorithm, HMM-KNN, that uses clusters based on execution progress similarity. In addition, we introduce a regression-based algorithm, HMM-GP, that uses Gaussian process regression to estimate the mean and standard deviation of the HMM's log-likelihood given the execution progress.



(a) Distribution of force and sound

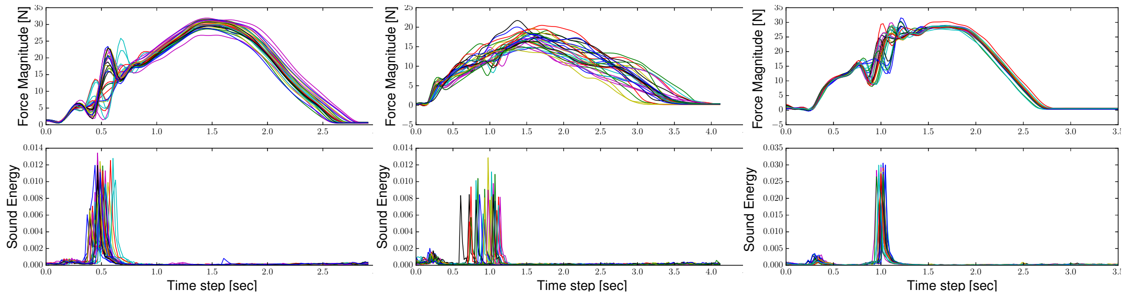


(b) Latching mechanism

Figure 3.3: Changes in the force and sound magnitudes during the execution of a microwave door closing task. Note that the force magnitude continues to increase even after latching since the operator, a PR2 robot, was programmed to push the door for a fixed duration of time.

3.2.1 Task-centric Multimodal Sensory Signals

Robots are able to access many sensory devices that may be informative during a task execution. We have observed haptic, auditory, visual, or kinematic sensory signals that are particularly relevant to manipulation tasks. Figure 3.3 shows an example of how two task-relevant signals, force and sound, change as a PR2 robot closes a microwave door. Depending on the state of the spring latch, associated forces and sounds vary. We can observe similar patterns of signals from pushing tasks with various objects, in which the robot repeatedly pushes with its end effector to close or turn on an object in a fixed amount of time. Figure 3.4 provides a visualization of force and sound changes recorded for 30 non-anomalous executions with three representative objects in 10 everyday object manipulation tasks (see Table 3.1). At any point, an anomaly can result in changes in a modality or multiple modalities. In particular, in Figure 3.4a, a small force anomaly may occur in the high variance area. Its detection will be non-trivial given only the force modality. We can complement the lack of information using other modalities such as the sound that co-occurs with the force.



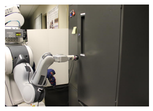
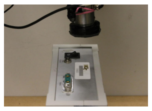
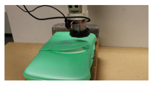
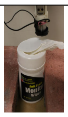
(a) Microwave (white) (b) Toolbox (c) Microwave (black)

Figure 3.4: Visualization of the force and sound sequences recorded in three representative manipulation tasks: closing microwave doors and latching a toolbox.

To model and monitor a task, we extract hand-engineered task-centric features from multimodal sensory signals. Other researchers have used automatic feature extraction methods for event detection or recognition using principal component analysis (PCA) [146] or autoencoders [147, 148]. However, anomalies do not necessarily come from the dominant components of observations. In other words, after the PCA, anomalies may not be detectable if their relevant signals were stationary in the training data. Instead, we use hand-engineered features based on our domain knowledge (see Section 3.2.4).

Our system represents sensory signals such as position and orientation with respect to a task, particularly an object. The task-centric representations are beneficial as they are pose-invariant from initial conditions of a PR2 and a target object. Inherently pose-invariant features such as sound and force are taken from raw input. We then extract hand-engineered features from raw sensory inputs. Note that after collecting raw sensory signals, we resample them to obtain the same sequence length since modalities have an imbalanced amount of information with distinct sensing frequencies. After extracting features $\mathbf{X}$, we scale each feature $\mathbf{x} \in \mathbf{X}$ individually to a given range, i.e., between 0 and 1, $\mathbf{x} = (\mathbf{x} - \mathbf{x}_{\min})/(\mathbf{x}_{\max} - \mathbf{x}_{\min})$.

Table 3.1: This table shows the objects the robot pushed in our experiments. The PR2 pushed each object while recording haptic and auditory data. The numbers in parentheses represent the number of non-anomalous and anomalous trials we conducted with each object.

Microwave Door	**Microwave Door**	**Cabinet Door**	**Light Switch**
(30, 6)	(30, 5)	 (30, 10)	(30, 10)
Device Switch  (31, 10)	**Outlet Switch** (30, 10)	**Toaster Switch** (33, 12)	**Diaper Case**  (30, 10)
Wipe Case (31, 11)		**Glasses Case** (32, 9)	

3.2.2 Generative Modeling

We formulate the problem of anomaly detection as the estimation of joint distribution $P(\mathbf{X}|\mathbf{X}_{\text{train}})$ given a set of training data $\mathbf{X}_{\text{train}}$ from normal activities, where $\mathbf{X}$ represents a sequence of multimodal observations (or task-centric features). When the estimated probability is lower than a threshold (see Section 3.2.3), our execution monitoring system determines $\mathbf{X}$ to be *anomalous*. In this section, we describe how to model the training data and list useful output for the estimation of a joint distribution.

Hidden Markov Models (HMMs)

We model extracted features using a left-to-right HMM with multivariate Gaussian emissions. Figure 3.5 depicts the architecture of the HMM and two examples of hidden-state paths (blue and red lines) associated with a time series of multidimensional observations. Let random variable $\mathbf{x}_i$ be an m-dimensional observation (i.e., feature) vector at time step

i. We represent a sequence of observations as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_i\}$. Let random variable $\mathbf{z}_i^j$ be the jth hidden state out of n hidden states at time step i. The left-to-right structure forces state indices to remain constant or increase over time. The transition probability $P(\mathbf{z}_{i+1}|\mathbf{z}_i)$ is the probability of transitioning from one hidden state $\mathbf{z}_i$ to another $\mathbf{z}_{i+1}$. The emission probability $P(\mathbf{x}_i|\mathbf{z}_i)$ is the probability of output $\mathbf{x}_i$ given a hidden state $\mathbf{z}_i$. To represent multi-dimensional output, we use a multivariate Gaussian distribution with a full covariance matrix so that we can represent correlations among modalities.

Collecting data across the entire space of anomalous executions is not feasible, so we use only negatively labeled *non-anomalous* data as training data. Assuming that a robot performs stereotyped task-specific behaviors, the training data $\mathbf{X}_{\text{train}}$ contains consistent patterns so that we can represent $\mathbf{X}_{\text{train}}$ as a set of model parameters λ_{MAP} through maximum a-posteriori (MAP) estimation. For λ_{MAP}, we use the Baum-Welch algorithm. We can approximate the estimation of the joint distribution as $P(\mathbf{X}|\lambda_{\text{MAP}})$. For convenience, we omit the subscript "MAP" when using parameter set λ.

Before training an HMM, we initialize the initial state distribution π and the transition probability matrix $\mathbf{A} \in \mathbb{R}^{\mathbf{n}\times\mathbf{n}}$ to use the left-to-right structure. We set the first element of the n-dimensional vector π to 1.0 and all other elements to 0.0 in order to start the HMM in the first state. We also set $\mathbf{A}$ to be an upper triangular matrix with linearly decreasing transition probabilities from 0.4 to 0.0 for gradual left-to-right state transitions. In this work, we use the General Hidden Markov Model library (GHMM) [149].

An HMM-induced Vector

While performing a task, an HMM can generate informative output such as likelihood, posterior probability distribution, or transition probabilities [40, 150]. We refer to a list of the outputs as an *HMM-induced vector*. The most commonly used output for classification is log-likelihood $l = \log P(\mathbf{X}|\lambda)$, which we can calculate from the sum of joint distributions, $P(\mathbf{X}|\lambda) = \sum_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z}|\lambda)$ [151], where $\mathbf{Z}$ is a state path over hidden state

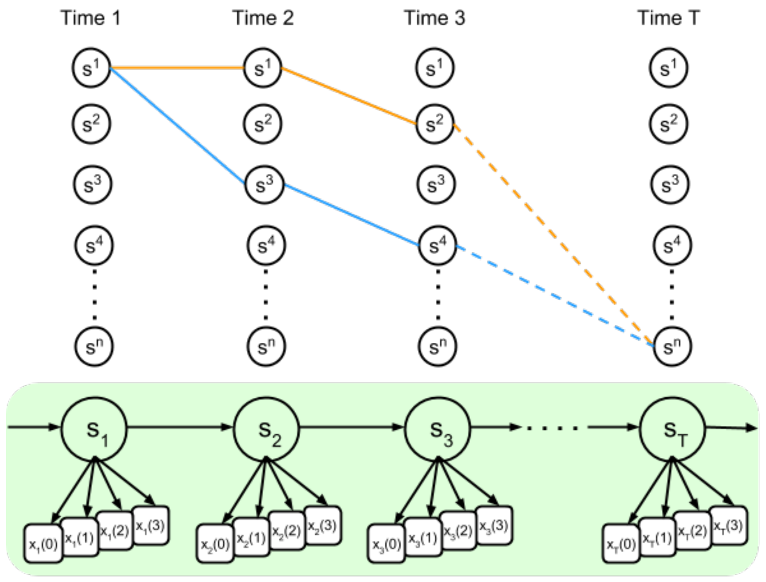


Figure 3.5: Architecture of left-right hidden Markov models with multivariate Gaussian emissions ($m = 4$).

space $\mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_T\}$ and T is the last time step in $\mathbf{X}$. Another output is the posterior probability distribution $P(\mathbf{z}_t|\mathbf{X}, \lambda)$, which we use as execution progress that represents the progress of a task-specific behavior at time t.

During non-anomalous executions, the likelihood tends to vary in consistent ways (see the upper graph in Figure 3.6) so we can model this variation with respect to the progress. The property of the left-to-right model is that hidden states must be in non-decreasing order such as $\mathbf{Z} = \{\mathbf{z}_1^1, \mathbf{z}_2^1, \mathbf{z}_3^2, \mathbf{z}_4^3, \mathbf{z}_5^4, ..., \mathbf{z}_T^n\}$[1]. If the true states were known, their indices could represent the progress. Compared to the direct use of time, the representation would have the advantage of handling variability in the timing of a behavior execution, but the true state path is hidden from the observer. One approach can represent the progress as the state indices with the maximum likelihood at any given moment, though the indices would neglect uncertainty. Instead, we represent the progress, $\gamma(t)$, as a probability mass function over hidden states (the hidden-state distribution) at time t, $\gamma(t) = P(\mathbf{z}_t|\mathbf{X}, \lambda)$. The lower graph in Figure 3.6 shows that the execution progress averaged across all non-anomalous trials can change in an intuitive way with respect to time with the index of the most likely

[1]The state path of an HMM always starts from the first hidden state, $\mathbf{z}^1$, setting $\pi = \{1, 0, ..., 0\}$

state progressively increasing. Note that HMMs will be able to generate a new dataset close to $\mathbf{X}_{\text{train}}$ given the execution progress but we do not investigate the generative properties in this paper. We compute the n-dimensional vector $\gamma(t)$ with the forward and backward procedures of the EM algorithm [144],

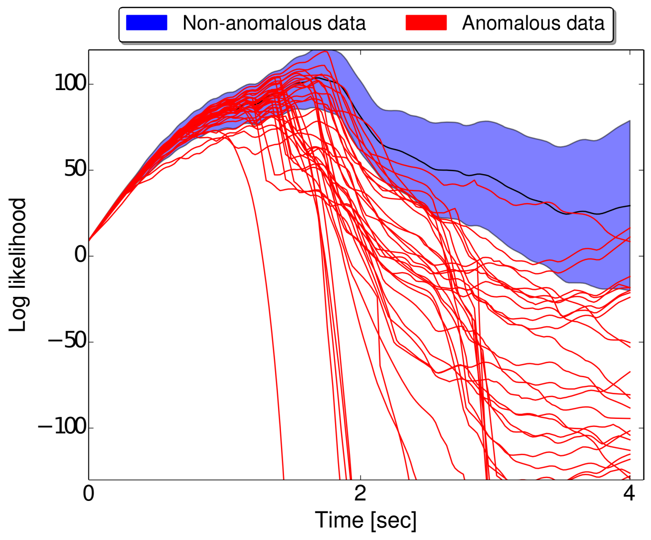
$$\gamma(t) = \frac{P(\mathbf{X}(1:t), \mathbf{z}_t \mid \lambda) \cdot P(\mathbf{X}(t+1:T) \mid \mathbf{z}_t, \lambda)}{P(\mathbf{X} \mid \lambda)}, \tag{3.1}$$

where T is the last time sample of $\mathbf{X}$.

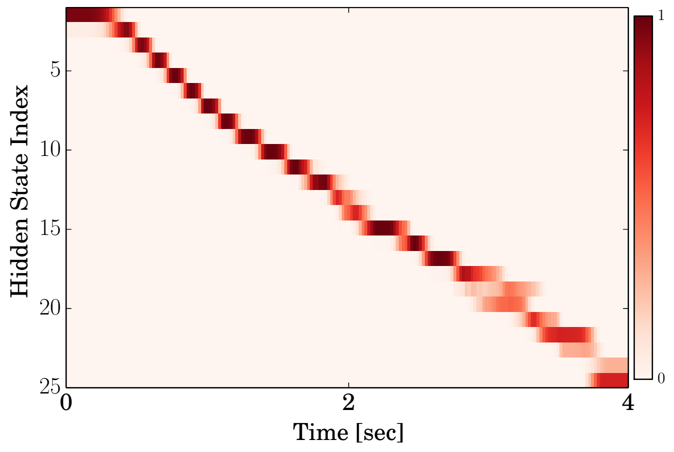
Figure 3.7 illustrates an example of execution progress and likelihood changes during a non-anomalous and an anomalous trial of a closing task. The upper two subgraphs show force and sound observations (blue curves) over time. Each white or green band represents a period of time over which the most likely hidden state remains constant. Each small black number indicates the index for the most likely hidden state over the duration of a band. These indices need not increase monotonically since we computed them in an online fashion using only prior observations at each time step. The blue curves of the bottom subgraphs show changes in the log-likelihoods, which we will discuss in Section 3.2.5.

3.2.3 Anomaly Detection

Our framework uses a binary, or one-class, classification method that learns from only non-anomalous execution data. It detects an anomaly when the log-likelihood of a sequence of observations X is lower than a varying threshold at some point in time during the execution. Otherwise, it considers the execution to be non-anomalous. We represent a threshold as $\hat{\mu}(\gamma) - c\hat{\sigma}(\gamma)$, where $\hat{\mu}$, $\hat{\sigma}$, and c are the expected log-likelihood, its standard deviation (or confidence interval), and a constant that adjusts the sensitivity of the classifier, respectively. For notational clarity, we omit t from $\gamma(t)$. In this section, we introduce two methods of estimating $\hat{\mu}$ and $\hat{\sigma}$ using clustering and regression methods.

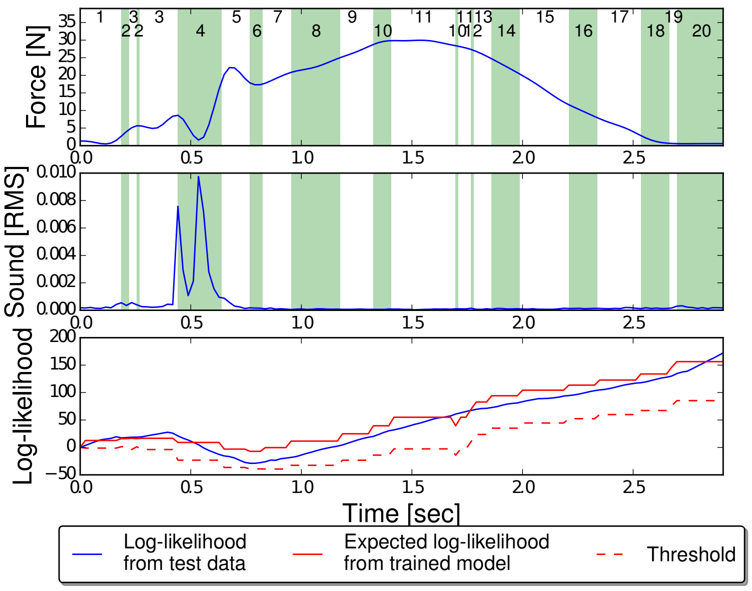


(a) Log-likelihood

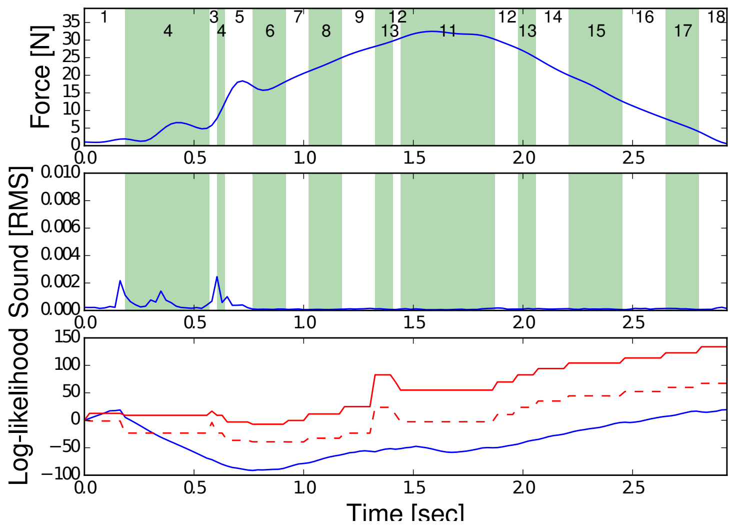


(b) Execution Progress

Figure 3.6: An example of log-likelihood and execution progress distributions (i.e., hidden-state distributions) when a PR2 robot closes the door of a microwave oven (white). (a) The blue-shaded region and red curves show the standard deviation of log-likelihoods from 35 non-anomalous executions and the unexpected drop of log-likelihoods from 18 anomalous executions, respectively. (b) This graph shows averaged changes of execution progress vectors from the non-anomalous executions.



(a) Non-anomalous operation



(b) Anomalous operation

Figure 3.7: Comparison of non-anomalous and anomalous observations in the door-closing task for a microwave (white). The upper two graphs for (a) and (b) show the force and sound observations over time. Each white or green band denotes a period of time over which the most likely hidden state remains constant. The small black number in each band is the index for the most likely hidden state over the duration of the band. These indices need not increase monotonically since we computed them in an online fashion using only previous observations at each time step. The bottom graphs in (a) and (b) illustrate the expected log-likelihood based on the execution progress (solid red curve), the log-likelihood resulting from the ongoing trial (solid blue curve), and the threshold based on execution progress (related to the dashed red curve). For this comparison, we set $c = 2.0$ and blocked the door using a rubber pad for the anomalous operation.

Clustering-based Threshold (HMM-D)

We first introduce a clustering-based method that provides a dynamically changing threshold. We call this method HMM-D that is an improved version from [42]. HMM-D clusters and parameterizes the HMM-induced vectors associated with only non-anomalous executions into K soft clusters. In the training step, we use Gaussian radial basis functions (RBFs) to produce the clusters. Assuming we have a similar phase of executions, we set evenly-distributed clusters over time that weight the membership of HMM-induced vectors based on RBFs. We then parameterize each cluster defined by a 3-tuple consisting of a weighted average of execution progress vectors, a weighted mean of log-likelihood vectors, a weighted standard deviation of log-likelihood vectors. The RBFs provide the weight. In the testing step, these clusters are then used to map execution progress vectors to estimates of the mean and standard deviation of the log-likelihood.

Figure 3.8 illustrates the distribution of RBFs, each of which weights execution progress vectors differently. For example, the kth cluster weights a vector at time t with its associated RBF $\phi(t, w_k) = e^{-\epsilon(t-w_k)^2}$, where $k \in \{1, ..., K\}$, w_k is the center of the kth RBF, and ϵ is a constant. Note that we use fixed length M of the time series in $\mathbf{X}_{\text{train}}$. We compute a weighted average of *execution progresses* $\hat{\gamma}_k$ at the kth cluster using

$$\hat{\gamma}_k = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{\eta_k} \sum_{t=1}^{T} \gamma^{(i)}(t) \phi(t, w_k) \right], \tag{3.2}$$

where N is the number of non-anomalous time series in $\mathbf{X}_{\text{train}}$, η_k is a normalization factor, $\eta_k = \sum_{t=1}^{T} \phi(t, w_k)$, and $\gamma^{(i)}(t)$ denotes the execution progress at time t for the ith time series ($i \in \{1, ..., N\}$).

Likewise, we also compute weighted mean $\hat{\mu}_k$ and variance $\hat{\sigma}_k$ of log-likelihoods at the

kth cluster,

$$\hat{\mu}_k(L) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{\eta_k}\sum_{t=1}^{T} L^{(i)}(t)\phi(t, w_k)\right], \tag{3.3}$$

$$\hat{\mu}_k(L \cdot L) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{\eta_k}\sum_{t=1}^{T} L^{(i)}(t)^2\phi(t, w_k)\right], \tag{3.4}$$

$$\hat{\sigma}_k(L) = \sqrt{\hat{\mu}_k(L \cdot L) - (\hat{\mu}_k(L))^2}, \tag{3.5}$$

where L denotes a set of log-likelihoods along non-anomalous time series,

$$L = \{L^{(1)}(0), ..., L^{(1)}(T), ..., L^{(N)}(0), ..., L^{(N)}(T)\}, \tag{3.6}$$

and $L^{(i)}(t)$ denotes a log-likelihood at time t for the ith time series. We parameterize and store the K clusters as

$$\{(\hat{\gamma}_1, \hat{\mu}_1, \hat{\sigma}_1), (\hat{\gamma}_2, \hat{\mu}_2, \hat{\sigma}_2), ..., (\hat{\gamma}_K, \hat{\mu}_K, \hat{\sigma}_K)\}, \tag{3.7}$$

where we omit L from $\hat{\mu}_k(L)$ and $\hat{\sigma}_k(L)$ for notional clarity.

At run time, we compute $\gamma(t)$ and $\log P(\mathbf{X}_t|\lambda)$ at time step t from the incoming signals. The detector then finds the index of the closest cluster, k^*, by comparing the difference between γ and each of the K clusters using symmetric Kullback-Leibler divergence (SKL-divergence),

$$k^* = \underset{1,...,K}{\arg\min}\ D_{\text{SKL}}(\gamma(t)||\gamma_k), \tag{3.8}$$

$$= \underset{1,...,K}{\arg\min}\ \min\left(D_{\text{KL}}(\gamma(t)||\gamma_k), D_{\text{KL}}(\gamma_k||\gamma(t))\right) \tag{3.9}$$

where $D_{\text{KL}}(P||Q)$ is a measure of the information lost when Q is used to approximate P.

The system determines an anomaly with following comparison:

$$\begin{cases} \text{anomaly} & \text{if } \log P(\mathbf{X}_t \mid \lambda) < \hat{\mu}_{k^*} - c\hat{\sigma}_{k^*} \\ \neg\text{anomaly}, & \text{otherwise} \end{cases} \tag{3.10}$$

where c is a real-valued constant to determine a sensitivity. Increasing c tends to result in fewer reported anomalies and an accompanying lower false positive rate and lower true positive rate. Decreasing c tends to result in more reported anomalies and an accompanying higher false positive rate and higher true positive rate.

In order to use HMM-D with training data from successful executions with large time variations, users would most likely need to align the data in time using dynamic time warping or another method. However, after training, anomaly detection would not require time-warped signals since the system only uses execution progress to estimate the mean and standard deviation of the log-likelihood. While we do not explicitly test tasks that involved large timing variations, our tasks have timing variations as illustrated in Figure 3.4. During the feeding task in Section 3.2.5, human actions also resulted in timing variations. In addition, to help clarify the role of the RBFs, we also introduce a state-based clustering method using k-nearest neighbors (k-NN) without using the RBFs. We call this method as HMM-KNN which will be discussed in Section 3.2.5.

Regression-based Threshold (HMM-GP)

We introduce an alternative detection method that uses regression to estimate the mean and standard deviation of the log-likelihood with respect to the execution progress. In the previous algorithms, the discontinuities between clusters may lower detection performance. To address this potential issue, we can apply any parametric or non-parametric regression method that provides mean and variance of the likelihood for our detector. In this dissertation, we use a Gaussian process regressor, referred to as GP, since it is also well supported by the community with open-source libraries and a number of variants [152, 153]. We se-

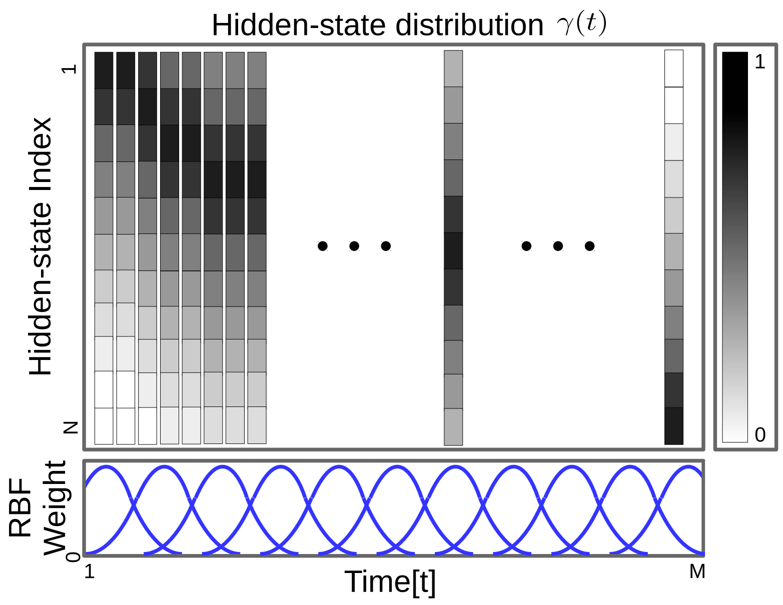


Figure 3.8: Illustration of the K clusters of execution progress vectors (i.e., hidden-state distributions). Each cluster has an associated RBF used to weight execution progress vectors $\gamma(t)$, and their associated log-likelihoods $L(t)$ based on when they occurred in time. These weights are used to compute $\hat{\gamma}_k$, $\hat{\mu}_k$, and $\hat{\sigma}_k$ for cluster k.

rially connect an HMM and a GP regression method that is able to output smooth detection thresholds with a comparably smaller amount of training data. It maps input and output pairs to predict an output given a new input with confidence intervals [154].

Let D be the length of an execution progress vector. Given training input and output pairs ($\boldsymbol{\gamma} \in \mathbb{R}^{NT \times D}$, $L \in \mathbb{R}^{NT}$), Gaussian process can model the predictive distribution of log-likelihood $l_* \in \mathbb{R}$ at an execution progress point $\gamma_* \in \mathbb{R}^D$ as

$$P(l_* \mid \gamma_*, L, \boldsymbol{\gamma}) = \mathcal{N}(\mu_*, \Sigma_*), \tag{3.11}$$

where μ_* and Σ_* are posterior mean and covariance estimates, respectively. In this work, we use a squared exponential function,

$$k(\gamma, \gamma') = \exp\left(-\frac{1}{2}\sum_{d=1}^{D} \frac{(\gamma(d) - \gamma'(d))^2}{l_d}\right) \tag{3.12}$$

to represent the covariance of $\mathbf{x}$ where l_d is an individual length scale hyperparameter for each input dimension d. We use a GP to find l_* that maximizes the marginal likelihood using

$$\mu_* = \mathbf{k}_*^T \mathbf{K}^{-1} L, \tag{3.13}$$

$$\Sigma_* = k_{**} - \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{k}_*, \tag{3.14}$$

where $\mathbf{K} = k(\boldsymbol{\gamma}, \boldsymbol{\gamma})$, $\mathbf{k}_* = k(\boldsymbol{\gamma}, \gamma_*)$, and $k_{**} = k(\gamma_*, \gamma_*)$. Through this process, we can find the output l_* with variance Σ_* given an observed execution progress.

To classify anomalies, HMM-GP also uses a statistical decision similar to the detection criteria described in Eq. (3.10):

$$\begin{cases} \text{anomaly} & \text{if } \log P(\mathbf{X}_t \mid \lambda) < \mu_* - c\sqrt{\Sigma_*} \\ \neg\text{anomaly}, & \text{otherwise} \end{cases} \tag{3.15}$$

where c is a constant that adjusts the sensitivity of the detector. Note that the covariance estimate Σ_* may have undergone undesired changes for each point by an ill-posed covariance matrix. To regularize it, we add constants, called nugget values [155], into the diagonal elements of the training-data covariance matrix to get smooth changes of the output distribution.

In terms of the amount of data, GP's non-parametric regression can be computationally expensive due to a large covariance matrix. In this section, we randomly subsampled training data to acquire a maximum of 1,000 samples to avoid the issue, and it performed well in Section 3.2.5. Other methods are also available to handle this issue, such as sparse Gaussian process (SPGP) [152].

3.2.4 Experimetal Setup

We evaluated our approach with pushing and robot-assisted feeding tasks selected from IADLs and ADLs. The pushing task includes behaviors such as closing doors and latching a toolbox. The feeding task includes scooping yogurt from a bowl and delivering spoonfuls of yogurt to the mouths of able-bodied participants and a person with disabilities (see Figure 3.11). We conducted this research with approval from the Georgia Tech Institutional Review Board (IRB), and obtained informed consent from all participants.

A Robotic Platform

Our system uses a PR2 robot from Willow Garage, a general-purpose mobile manipulator with two 7-DOF back-driveable arms with powered grippers and an omni-directional mobile base. For safety and prevention of possible hazards, we used a $1\,\mathrm{kHz}$ low-level PID controller with low gains and a $50\,\mathrm{Hz}$ mid-level model predictive controller [156] without haptic feedback. Its maximum payload and grip force are listed as $1.8\,\mathrm{kg}$ and $80\,\mathrm{N}$, respectively. We made the PR2 hold a tool and mounted sensors on it.

Instrumentation for Multimodal Sensing

For the pushing and scooping behaviors, the PR2 held an instrumented tool with a 3D-printed handle designed for its grippers (see Figure 3.9). The tool included a force/torque sensor (ATI Nano25) and a unidirectional microphone for monitoring haptic and auditory signals. During manipulation, our monitoring system recorded 6-axis force/torque measurements at $1\,\mathrm{kHz}$, and simultaneously recorded sound input from the microphone at $44.1\,\mathrm{kHz}$. We also used two additional cross-modal features, relative position and relative orientation between the tool and a target object. The system determined its tool pose using joint encoder values and forward kinematics. It estimated a target pose using a Microsoft Kinect V2 and an ARTag [157] attached to the door of the microwave oven at $10\,\mathrm{kHz}$.

For the feeding behavior, the PR2 also held a tool with a force/torque sensor. To measure a person's mouth pose as a target, we used an RGB-D camera (Intel SR300) mounted on the wrist of one arm (see Figure 3.10).

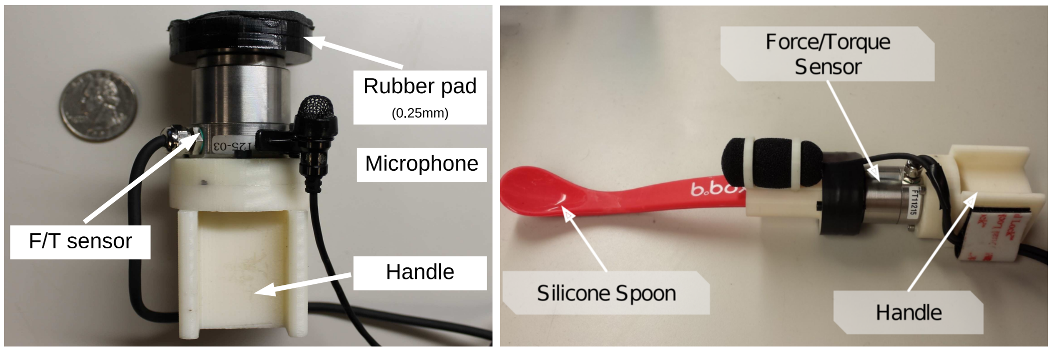


Figure 3.9: Each instrumented tool for pushing and scooping tasks has a force-torque sensor and a microphone mounted on a 3D-printed handle, designed to be held by the PR2 gripper. **Left:** A pushing tool with a rubber-padded plastic circle. **Right:** A scooping tool with a flexible silicone spoon.

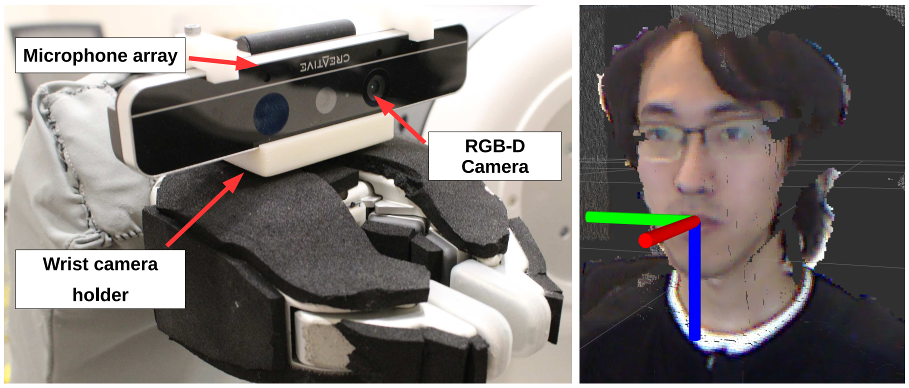


Figure 3.10: A wrist-mounted sensing tool for the feeding task. **Left:** An RGB-D camera and a microphone array. **Right:** Mouth pose detection using the point cloud of the camera.

Feature Selection and Preprocessing

We extracted hand-engineered, task-centric features from multimodal sensory signals to train a trans-invariant detector since the coordination between a robot and a target varies from execution to execution. Based on our domain knowledge, we extracted the following features over time:

- **Pushing behavior**: *sound*(s), *force*(f), *approach distance*(k), and *approach angle*(k)[2]. *Sound* is audio energy expressed in Eq. (3.16). *Force* is the magnitude of force $\mathbf{f}$ on the end-effector ($f = \|\mathbf{f}\|_2$). *Approach distance* is the Euclidean distance between the

[2]The symbols, f, s, and k, in the parentheses represent force, sound, and kinematic modalities, respectively.

tool and an ARTag attached on a target object. *Approach angle* is the angular difference between the pushing tool of the robot and a line perpendicular to a door or lid plane.

- **Scooping behavior**: *sound*(*s*), *force*(*f*), *approach distance*(*k*), and *approach angle*(*k*). *Approach distance* is the distance between the tool and the bottom of a bowl location estimated by the kinematics of the robot. *Approach angle* is the angular difference between the tool and a line perpendicular to the opening plane of the bowl.

- **Feeding behavior**: *sound*(*s*), *force*(*f*), *joint torque*(*f*), and *approach distance*(*k*). *Force* is the directional component of the force vector along the spoon. *Joint torque* is the feedback torque of the first pan-joint, which corresponds to rotation of shoulder around the vertical axis, of the PR2.

The feature we used should be applicable to a wide variety of tasks. We clarified the correspondence of the features through modality analysis in Sec 3.2.5.

To extract audio energy, we use the "Yaafe audio feature extraction toolbox" [158] to convert a frame s into a numeric value of energy $\mathcal{E}$ using the root mean square (RMS),

$$\mathcal{E} = \sqrt{\frac{\sum_{i=1}^{N_{\text{frame}}} \left(s(i)/I_{\max}\right)^2}{N_{\text{frame}}}}, \tag{3.16}$$

where N_{frame} is the audio frame size, which varies from 1,024 to 4,096 and $I_{\max}$ is 32,768—the maximum value of a 16-bit signed integer format.

To extract the *approach distance* and *approach angle* in the feeding behavior, the robot continuously tracks a user's mouth using the wrist-mounted RGB-D camera. After locating 2D facial landmarks using the dlib library [159, 160], we converted landmarks into 3D based on depth information and estimated the position and orientation of the mouth, similar to the process in [5]. We calculated *approach distance* by finding the Euclidean distance between the target and end-effector position. We calculated *approach angle* that is angular difference $2\arccos(q_1, q_2)$, where q_1 and q_2 are the orientations (i.e., quaternions) of the target and the end-effector, respectively.

Algorithm 1 shows how our system preprocesses the extracted feature vectors to be identical in scale, length, and offset. For each feature of data, let $\mathbf{V} \in \mathbb{R}^{N \times M_{\text{raw}}}$ be a set of feature vectors from both non-anomalous and anomalous executions, where N and M_{raw} are the number of samples and length, respectively. We first zero each feature vector $\mathbf{v} \in \mathbf{V}$ by subtracting the average of the first four elements, $\mathbf{v} = \mathbf{v} - \sum_{i=1}^{4} \mathbf{v}[i]/4$. This zeroing makes $\mathbf{v}$ starts from the same value as other vectors regardless calibration or white noise. Then, we scale each $\mathbf{v}$ between 0 and 1. To handle different sampling rates and match with actual anomaly check frequency in the PR2, we fit $\mathbf{v}$ with a B-Spline curve and resample it to a length M. In this section, we used $M = 140$ for the robot-assisted feeding and 200 otherwise. This approximately matches the rate at which our execution monitor runs in real time. Finally, this preprocessing resulted in a sequence of tuples. For example, in the case of *sound* $\mathcal{E}$ and *force* f, the sequence of length t was $\{(f_1, \mathcal{E}_1), (f_2, \mathcal{E}_2), ..., (f_t, \mathcal{E}_t)\}$.

Algorithm 1: Preprocessing

```
input  : $\mathbf{V} \in \mathbb{R}^{N \times M_{\text{raw}}}, M$
output: $\mathbf{V} \in \mathbb{R}^{N \times M}$
1 $v_{\max} \leftarrow \max(\mathbf{V}), v_{\min} \leftarrow \min(\mathbf{V})$;
2 for $i \leftarrow 1$ to $N$ do
3     $\mathbf{v} \leftarrow \mathbf{V}[i] - \frac{\sum_{i=1}^{4} \mathbf{V}[i]}{4}$ ;                    // Zeroing
4     $\mathbf{v} \leftarrow \frac{\mathbf{v} - v_{\min}}{v_{\max} - v_{\min}}$ ;                    // Scaling
5     $\mathbf{v} \leftarrow \text{resample}(\text{curveFitting}(\mathbf{v}), M)$ ;      // Sampling
6     $\mathbf{V}[i] \leftarrow \mathbf{v}$;
7 end
```

Experiment and Evaluation Procedures

For our experiments, we defined an anomalous execution as one that was inconsistent with typical executions. For example, even if the robot did scoop the yogurt in the scooping task, we classified the execution as anomalous if the user loudly commanded the robot to stop.

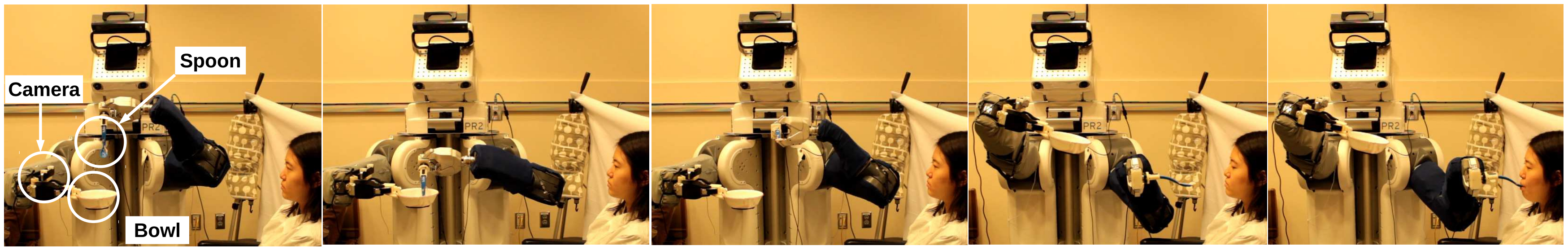
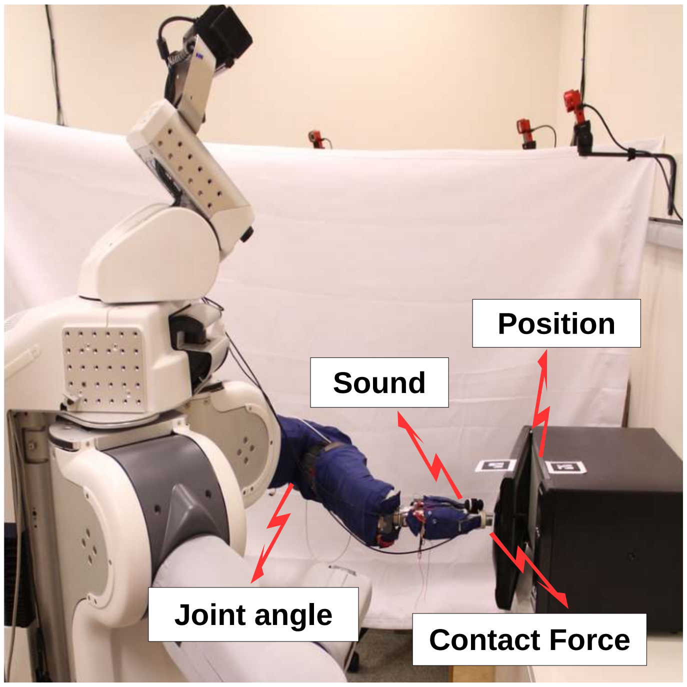


Figure 3.11: The images show the entire scooping and feeding process with an able-bodied participant.

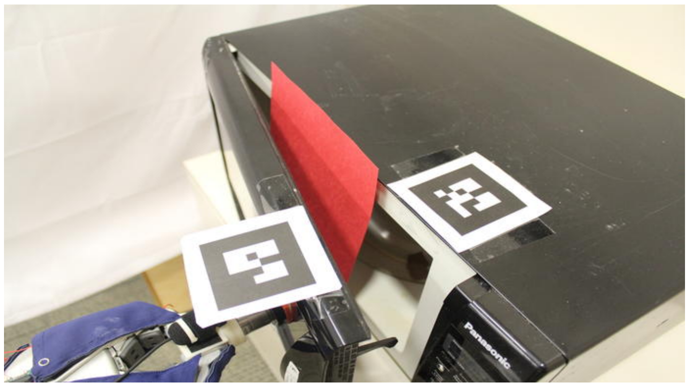
Pushing Task We collected pushing task data from two microwave ovens and a toolbox (see Figure 3.12). The PR2 pushed each object with the instrumented tool and a predefined linear end-effector trajectory for an object-specific amount of time and then pulled its end effector back. To produce anomalous events, we placed the tool at an incorrect location from which it could not properly contact the target mechanism, fixed the mechanism to prevent movement, or blocked the mechanism using an obstacle such as a stack of paper, a fabric, a rubber pad, a ceramic bowl or a finger. During each execution, applied forces, sounds, end-effector poses, and object poses were measured via the sensing tools, the Kinect camera and the joint encoders. We recorded 54, 52, and 53 executions with non-anomalous data and 55, 56, and 51 executions with anomalous data in microwave white (Microwave (W)), microwave black (Microwave (B)), and toolbox closing tasks, respectively. The anomalous executions also included unintended failures while performing non-anomalous executions.

Robot-assisted Feeding Task We collected data from a robot-assisted feeding task for which the PR2 executed a sequence of movements to scoop a spoon of yogurt from a bowl and then feed the yogurt to a human (see Figure 3.11). For the scooping data collection, we recruited 5 able-bodied participants. We briefly trained them to use the system with one or two trials and then asked them to freely use the scooping and feeding behaviors in which they determined *non-anomalous* and *anomalous* labels for each behavior. We recorded 72 non-anomalous and 45 anomalous executions for the scooping behavior. To create anomalous executions, we asked each participant to intentionally fail when scooping yogurt in the following way:

- By having each participant push any part of the spoon, the bowl, or the arm of the PR2 during the scooping process
- By having the participant yell anything at any moment.



(a) A scene of pushing experiment


(b) A paper obstacle

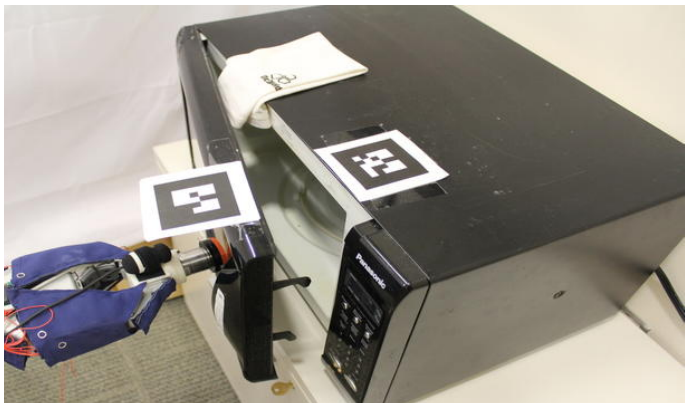

(c) A fabric obstacle

Figure 3.12: Sample scenes of pushing experiment with a microwave (B) door. (b) and (c) show two produced anomalies with paper and fabric obstacles, respectively.

For the feeding data collection, we recruited 8 able-bodied participants [3] and collected data from 192 anomalous feeding execution (=12 types $\times$ 2 executions per type $\times$ 8 participants) as well as 160 non-anomalous feeding execution. During non-anomalous feeding, we asked participants not to move their upper bodies except their necks to simulate the feeding task with motor impairments. To collect a broad range of anomalies, we selected 12 representative causes (see Figure 3.13) from three groups of anomalies described in the literature [17, 18],

- Human Error: a spoon miss, a spoon collision, a robot-body collision, aggressive eating, an anomalous sound, an unreachable mouth pose, and face occlusion.
- Engineering Error: a spoon miss, a spoon collision, and a controller failure.
- Environmental Error: an obstacle collision and a noise.

For each cause of anomalies listed, we briefly explained and showed a demonstration video, then we asked the participants to reproduce the anomaly twice at any time, magnitude, and direction during robot-assisted feeding.

Evaluation Setup

Methods In addition to our two proposed methods,

- HMM-D: A likelihood-based classifier with a clustering-based threshold.
- HMM-GP: A likelihood-based classifier with a regression-based threshold.

We implemented 6 base-line methods,

- RANDOM: This method randomly determines the existence of anomalies.
- OSVM: This method is an SVM-based detector trained with only negative training data (i.e., successful executions). Similar to [37], we move a sliding window (of size 10 in

[3]Participants were 3 males and 5 females. Their age ranges from 19 to 35. They are either attending or have graduated college.

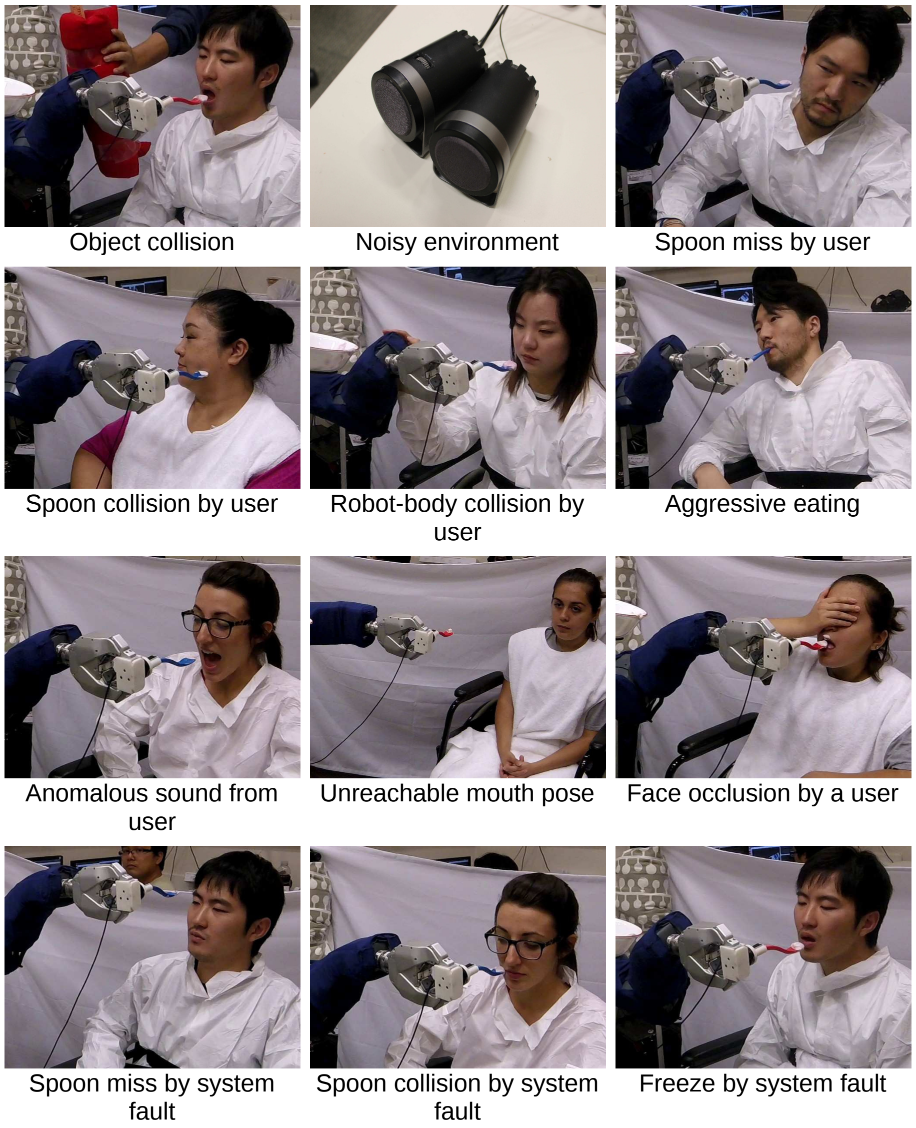


Figure 3.13: 12 representative cases for each anomalies in assistive feeding task with able-bodied participants.

time) one step at a time over length of 140 or 200. After flattening and normalizing it with a zero mean and unit variance, we input the data to an OSVM with an RBF (Gaussian) kernel. We control its sensitivity by adjusting the number of support vectors.

- HMM-OSVM: We combined an HMM with an OSVM similar to [161]. Instead of the sliding window, we input an HMM-induced vector to the OSVM at each time step.
- HMM-C: A likelihood-based classifier that detects an anomaly when the change in log-likelihood exceeds a fixed threshold [103].
- HMM-F: A likelihood-based classifier with a fixed threshold [101].
- HMM-KNN: A likelihood-based classifier with a clustering-based threshold. Unlike HMM-D, this method uses k-nearest neighbors to create clusters instead of RBFs.

Cross Validation For the pushing and scooping behaviors, we performed k-fold cross-validation, in which we randomly split non-anomalous and anomalous data into k folds, independently. To form the test data, we paired one fold from the non-anomalous data and one from the anomalous data, and used the remaining $k - 1$ folds of the non-anomalous data for training. We repeated this process k^2 times so that each possible pair was used exactly once as test data. Note that we used $k = 3$ in this work. For the feeding behavior, we used leave-one-person-out cross validation for multiple subject data.

Training consisted of first fitting an HMM to the specific behavior and the particular object or human user. Using the output of the HMM, we then trained each classification method. In this study, we fixed the number of hidden states at 25 that is empirically determined and applied to all HMMs used in this section. For the progress-based classifier, we used 25 RBFs, to match the number of hidden states. We added small random noise into the training data to provide variance.

Table 3.2: Detection performance (AUC) of our two methods and six baseline methods with 4 hand-engineered features from force, sound, and kinematic modalities. We show the highest AUC for each task in bold. Each task is categorized as either instrumental activities of daily life (IADL) or activities of daily life (ADL).

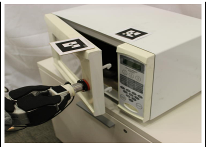
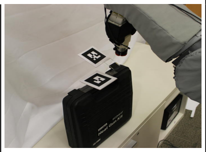
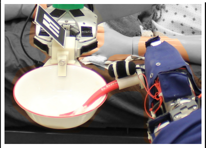
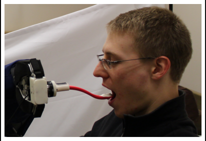
Method	IADL			ADL		Average
	Microwave (B)	 Microwave (W)	 Toolbox	 Scooping	 Feeding	
Random	0.4782	0.5001	0.4454	0.4750	0.5121	0.4822
OSVM	0.5023	0.6589	0.4277	0.6870	0.7238	0.5999
HMM-F	0.7764	0.6084	0.6735	0.6038	0.5055	0.6335
HMM-OSVM	0.7439	0.6436	0.6655	0.6478	0.7179	0.6837
HMM-C	0.8828	0.9405	0.4454	0.7183	0.7711	0.7516
HMM-KNN	0.9057	0.9853	0.7402	0.8249	0.8015	0.8515
HMM-D	0.9072	0.9935	**0.7796**	0.8293	0.8079	0.8627
HMM-GP	**0.9113**	**0.9938**	0.7654	**0.8608**	**0.8121**	**0.8687**

Table 3.3: Detection performance over modalities with HMM-GP. Modalities *f*, *s*, and *k* represent force, sound, and kinematic modalities, respectively. (e.g., *(f,s,k)* indicates the use of all modalities used in Table 3.2).

Modalities	Microwave (B)	Microwave (W)	Toolbox	Scooping	Feeding	Average
s	0.6839	0.7353	0.5496	0.5168	0.6225	0.6216
f	0.5058	0.7024	0.4686	0.8123	0.6441	0.6266
f,s	0.6672	0.8288	0.5537	0.7178	0.7632	0.7061
k	0.7703	0.8199	0.6193	0.8212	0.6355	0.7332
s,k	0.8922	0.9759	0.6524	0.7999	0.6173	0.7875
f,k	0.8665	0.9218	0.6743	**0.9568**	0.7458	0.8330
f,s,k	**0.9113**	**0.9938**	**0.7654**	0.8608	**0.8121**	**0.8687**

3.2.5 Evaluation and Discussion

Examples of Time-varying Thresholds

The bottom graphs of Figure 3.7 show our time-varying likelihood threshold from the clustering method. For the non-anomalous execution, the mean log-likelihood based on execution progress (solid red curve) moves in conjunction with the log-likelihood resulting from the ongoing trial (solid blue curve). The standard deviation based on execution progress (related to the dashed red curve) tends to increase over time. For the anomalous execution, the behavior failed to generate a sharp sound at the appropriate time and instead generated a lower magnitude sharp sound early in the behavior's execution in conjunction with lower forces than anticipated. As a result, the index 4 was not likely transitioned to a next index compared to the non-anomalous execution. The log-likelihood went below the threshold early on in the execution, triggering the detection of an anomaly. This illustrates that execution progress is helpful to determine a tighter decision boundary than a fixed threshold.

Performance Comparison with Baseline Methods

AUC We compared our HMM-D and HMM-GP to baseline methods. Table 3.2 shows HMM-GP resulted in higher AUC than others over the five tasks. HMM-GP showed the highest AUC in four tasks with a maximum of 0.9938. HMM-D shows similar but slightly lower performance, but it outperformed HMM-KNN. This result shows the training with the time-based RBFs was beneficial. In addition, the feeding task's AUCs, computed by the leave-one-person-out cross validation, demonstrates our that an execution monitor can generalize to new people. In Figure 3.14, the receiver operating characteristic (ROC) curves also show HMM-GP and HMM-D result in higher true positive rates (TPR) than the other baseline methods for comparable false positive rates (FPR). The performance was high with the microwave task. However, the feeding task performance was highly dependent on the types of anomalies. To illustrate this, Figure 3.14 shows an ROC curve with all 12 anomalies and an ROC curve that only considers the 6 best detected anomalies. In general, this highlights the importance of using modalities and features suitable for detecting important types of anomalies. In our evaluation, we attempted to use a wide variety of realistic anomalies, some of which were challenging for the system.

Another interesting result is that OSVM showed lower performance than others except RANDOM. The reason could be that OSVM cannot sufficiently convey contextual information from the past since it uses a sliding window. HMM-OSVM's performance was substantially better than OSVM, but its performance was still lower than our proposed methods. The reason could be that the highly nonlinear RBF kernel in HMM-OSVM led to overfitting and generated false alarms. In addition, 6 out of 8 methods we evaluated had their worst performance (AUC) on the toolbox pushing task, and no method performed particularly well. This is most likely due to the high variability in the signals (see Figure 3.4b). The especially poor performance of HMM-C suggests that the likelihood values produced by the HMM could exhibit large changes from time step to time step in the toolbox pushing task. Note that rounding made the AUC rates of RND and HMM-C appear the same. The raw AUC rates of RND and HMM-C were 0.445410449295 and 0.445389896, respectively.

Through evaluation, in terms of the amount of data we had, HMM-GP was well matched to the pushing tasks, which had 50 or fewer time series when considering cross validation. For the feeding task, HMM-GP was not as well matched due to the larger amount of data. Consequently, we performed random sub-sampling of the data, which performed well in practice.

Detection Delay Another performance measure for an execution monitor is the detection delay. Figure 3.15 presents a comparison between the detection delay and the true positive rate changes of our methods and the HMM-F method in the feeding task. We generated various simulated anomaly data by adding a step signal from a magnitude of 1% to 150% of the highest real anomaly signals to a feature of the non-anomalous data at random times.

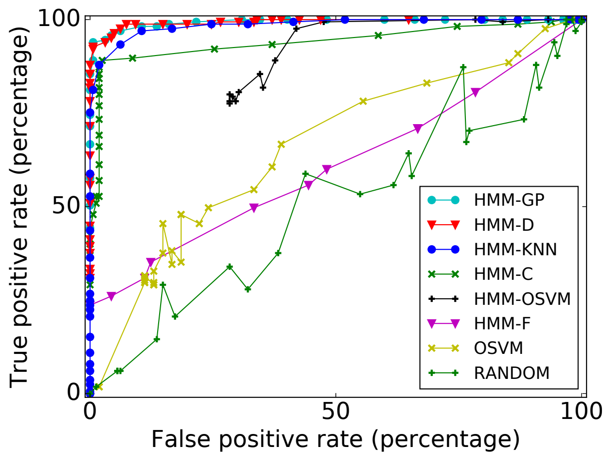
The simulated anomaly data corresponds to "spoon collision," "arm collision," "spoon miss by a user," and "anomalous sound." The HMM-D and -GP methods resulted in shorter detection delays with higher true positive rates. In particular, our methods detected small anomalies ($\leq 10\%$) within 3.0 second. On the other hand, HMM-F was able to detect the large anomalies ($>10\%$) with longer detection delays and lower true positive rates (TPR). In this process, the higher sensitivity (i.e., TPR) we used to select the parameter of classifiers, the shorter the detection delay we observed.

In this evaluation, we tuned the three detectors to have the same, small false positive rate (FPR) given a test set. Similar to Neyman-Pearson criteria [162], we bound the range of false positive rates and maximized the true positive rates. However, due to limited data, the detectors could not achieve arbitrary false positive rates. Consequently, we exhaustively searched the false positive rates achievable by all three detectors. We found that $1.0 \pm 1.0\%$ is the range of the commonly achievable smallest false positive rate. Note that we did not evaluate other detector tunings, so we can not be certain that these results would hold for other tunings. However, we would expect the general trend of longer detection delays for more subtle anomalies to hold.

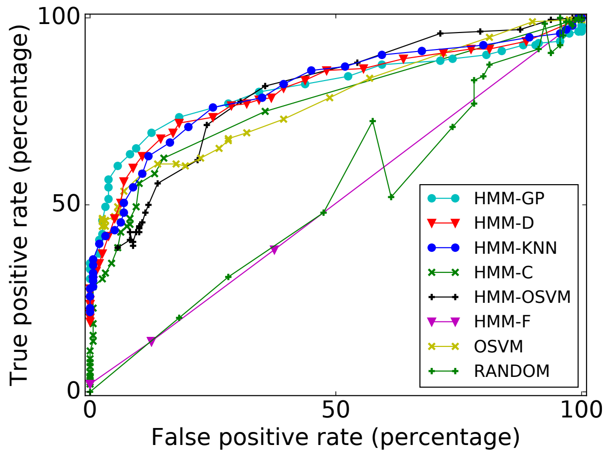
Multimodality

Multimodality vs. Unimodality We investigated if multiple modalities improve the detection performance of our monitoring system. Table 3.3 shows AUC over combinations of modalities (i.e., force, sound, kinematics [4]) with HMM-GP. The use of all three modalities achieved the best performance in four of the five tasks. The use of one, two, or three modalities showed maximum 0.7332, 0.8830, and 0.8687 of the averaged AUC, respectively. The AUC indicated that multi-modalities were substantially more effective than unimodalities in this detection. However, it does not indicate that more modalities always enhance detection performance. In the scooping task, including sound as a modality re-

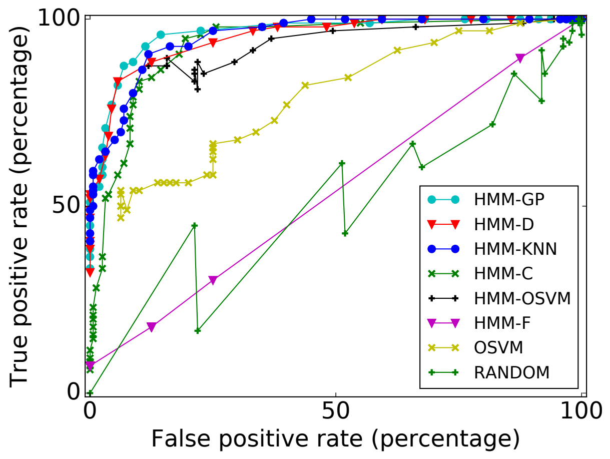
[4]Kinematic modality refers to the task-kinematic input measured by the encoder and vision sensors (i.e, relative distance or orientation between the PR2 and a target object or human).



(a) Pushing task with the microwave (W)

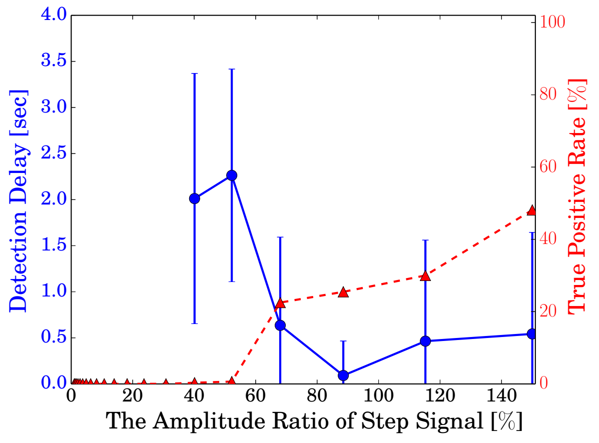


(b) Feeding task with 12 anomalies

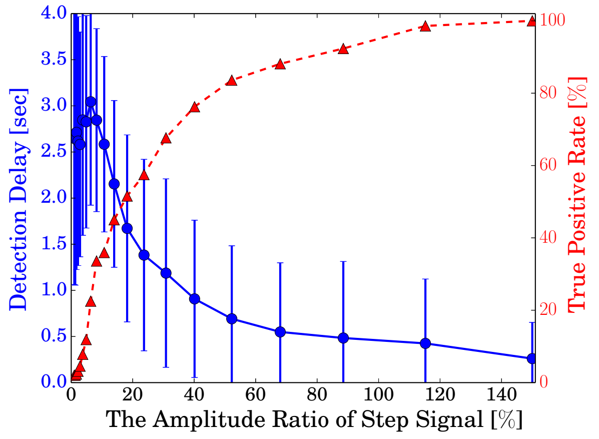


(c) Feeding task with 6 anomalies selected from the top of Figure 3.16

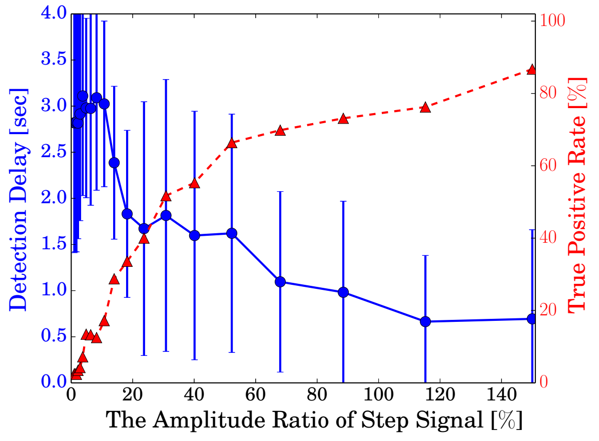
Figure 3.14: Receiver operating characteristic (ROC) curves to evaluate the performance of our anomaly detectors versus six baseline detectors in the pushing task with the microwave (W) and the feeding task.



(a) HMM-F



(b) HMM-D



(c) HMM-GP

Figure 3.15: Comparison between the detection delays and the true positive rates on simulated anomaly signals introduced in the assistive feeding task. We added a step signal to a feature of non-anomalous data in which the amplitude of a signal ranges from 1% to 150% of the maximum amplitude of observed anomaly signals in real experiments. We take only positive delays into account.

duced the detector's performance. It seems likely that the sound recordings for this task varied in a manner that was unrelated to the anomalies we tested, and thus did not improve anomaly detection and instead resulted in false positives.

Effective Modalities Effectiveness of modalities varies according to the type of anomaly. Figure 3.16 shows the TPR matrix in which the x- and y-axes represent input modalities and tested anomaly classes, respectively. We ordered anomaly classes with respect to a similar TPR distribution over the three unimodalities using density-based spatial clustering of applications with noise (DBSCAN). The clustering resulted in four groups of anomalies (red brackets and numbers). The first three groups show that the detection of anomalies primarily depends on a single modality. We also observed that multimodality improved detection performance. For example, the force-kinematics modality combination yielded higher TPR in three of four anomalies in the first group. All three combinations of modalities also enabled our system to detect the broadest range of anomalies. However, the anomalies in the last group were not easily detectable. In particular, "face occlusion" and "freeze" were hardly detectable from any combination of modalities that we used. We expect that additional features or modalities might help us to detect these anomalies. For example, modalities that monitor the internal state of the robot, including the activity of computational processes, might better detect some types of system freezes and faults. Note that we only produced these results for specific detector tunings, so the details could change with other tunings. However, we would expect the associations between particular modalities and particular types of anomalies to hold across tunings, as well as the existence of anomalies that are difficult to detect even with all of the modalities available.

3.3 Multimodal Anomaly Detection II: LSTM-VAE

Our previous work used also selected 4 hand-engineered features from 3 modalities for a likelihood-based classifier, HMM-GP, using hidden Markov models (HMM) [42, 163].

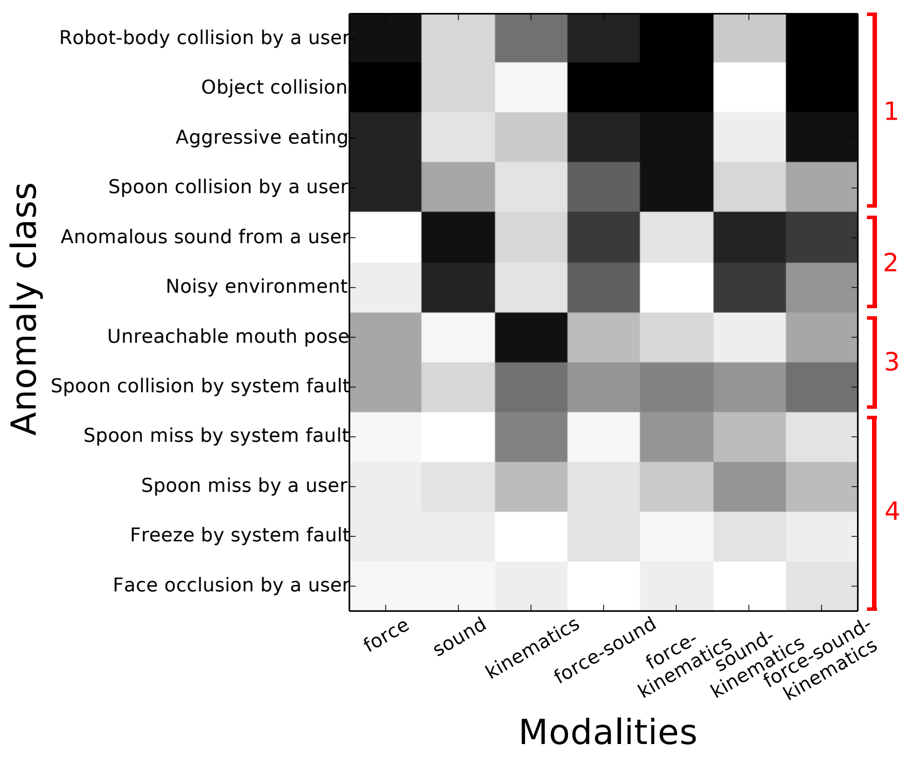


Figure 3.16: True positive rate per anomaly class and modalities in the assistive feeding task. Each column shows the true positive rate distribution of an HMM-GP from a combination of modalities. We tuned the threshold of the detector to produce an $8 \pm 0.7\%$ false positive rate (FPR) that is the commonly achievable smallest value by the 7 set of modalities. Black and white regions represent 100% and 0% true positive rates, respectively. Red brackets and numbers show four clusters found via DBSCAN.

However, the compressed or selected representations may be missing information relevant to anomaly detection. Creating useful hand-specified features can also involve significant engineering effort and domain expertise.

An alternative solution is reconstruction-based detection, such as an autoencoder (AE) based approach that compresses and reconstructs high dimensional inputs based on non-anomalous executions. When an AE is trained only with non-anomalous data, a high reconstruction error can indicate an anomaly. The idea behind this detection is that an AE cannot reconstruct unforeseen patterns of anomalous data well compared to foreseen non-anomalous data. In addition to the reconstruction error, a variational autoencoder (VAE) can compute the reconstruction log-likelihood of the inputs modeling the underlying probability distribution of data. Both AE and VAE can be combined with time-series modeling

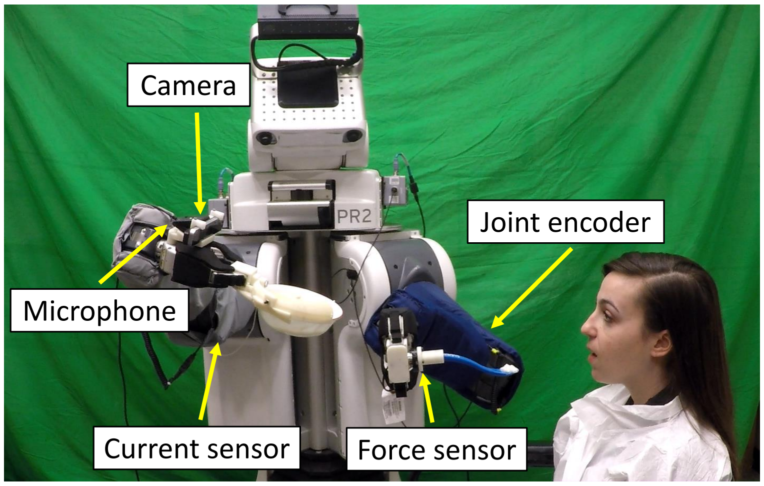


Figure 3.17: Robot-assisted feeding system. A PR2 robot detects anomalous feeding executions collecting 17 sensory signals from 5 types of sensors.

approaches such as recurrent neural network (RNN) including long short-term memory (LSTM) network.

In this section, we introduce a long short-term memory-based variational autoencoder (LSTM-VAE) for multimodal anomaly detection [164]. For encoding, an LSTM-VAE projects multimodal observations and their temporal dependencies at each time step into a latent space using serially connected LSTM and VAE layers. For decoding, it estimates the expected distribution of the multimodal inputs from the latent space representation. We train it under a denoising autoencoding criterion [143] to prevent learning an identity function and improve representation capability. Our LSTM-VAE-based detector detects an anomaly when the log-likelihood of current observation given the expected distribution is lower than a threshold. We also introduce a state-based threshold to increase detection sensitivity and lower the false alarms similar to [42].

3.3.1 Preliminaries

We review an autoencoder and a variational autoencoder. We represent a vector of multidimensional inputs by $\mathbf{x} \in \mathbb{R}^D$ and the corresponding latent space vector by $\mathbf{z} \in \mathcal{R}^K$, where D and K are the number of input signals and the dimension of the latent space, respectively.

Autoencoder(AE)

An AE is an artificial neural network that consists of sequentially connected encoder and decoder networks. It sets the target of the decoder to be equal to the input of the encoder. The encoder network learns a compressed representation (i.e., bottleneck feature or latent variable) of the input. The decoder network reconstructs the target from the compressed representation. The difference between the input and the reconstructed input is the reconstruction error. During training, the autoencoder minimizes the reconstruction error as an objective function. An AE is often used for data generation as a generative model. An AE's decoder can generate an output given an artificially assigned compressed representation.

Variational Autoencoder(VAE)

A VAE is a variant of an AE rooted in Bayesian inference [83]. A VAE is able to model the underlying distribution of observations $p(\mathbf{z})$ and generate new data by introducing a set of latent random variables $\mathbf{z}$. We can represent the process as $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$. However, the marginalization is computationally intractable since the search space of $\mathbf{z}$ is continuous and combinatorially large. Instead, we can represent the marginal log-likelihood of an individual point as $\log p(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \mathcal{L}_{\text{vae}}(\phi, \theta; \mathbf{x})$ using notation from [83], where D_{KL} is Kullback-Leibler divergence from a prior $p_\theta(\mathbf{z})$ to the variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ of $p(\mathbf{z}|\mathbf{x})$ and $\mathcal{L}_{\text{vae}}$ is the variational lower bound of the data $\mathbf{x}$ by Jensen's inequality. Note that ϕ and θ are the parameters of the encoder and the decoder, respectively.

A VAE optimizes the parameters, ϕ and θ, by maximizing the lower bound of the log likelihood, $\mathcal{L}_{\text{vae}}$,

$$\mathcal{L}_{\text{vae}} = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]. \tag{3.17}$$

The first term regularizes the latent variable $\mathbf{z}$ by minimizing the KL divergence between the approximated posterior and the prior of the latent variable. The second term is the reconstruction of $\mathbf{x}$ by maximizing the log-likelihood $\log p_\theta(\mathbf{x}|\mathbf{z})$ with sampling from $q_\phi(\mathbf{z}|\mathbf{x})$.

The choice of distribution types is important since a VAE models the approximated posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ from a prior $p_\theta(\mathbf{z})$ and likelihood $p_\theta(\mathbf{x}|\mathbf{z})$. A typical choice for the posterior is a Gaussian distribution, $\mathcal{N}(\mu_\mathbf{z}, \Sigma_\mathbf{z})$, where a standard normal distribution $\mathcal{N}(0, 1)$ is used for the prior. For the likelihood, a Bernoulli distribution or multivariate Gaussian distribution is often used for binary or continuous data, respectively.

3.3.2 An LSTM-based Variational Autoencoder (LSTM-VAE)

We present a long short-term memory-based variational autoencoder (LSTM-VAE). To introduce the temporal dependency of time-series data into a VAE, we combine a VAE with LSTMs by replacing the feed-forward network in a VAE to LSTMs similar to conventional temporal AEs such as an RNN Encoder-Decoder [81] and an EncDec-AD [80]. Fig. 3.18 shows an unrolled structure with LSTM-based encoder-and-decoder modules. Given a multimodal input $\mathbf{x}_t$ at time t, the encoder approximates the posterior $p(\mathbf{z}_t|\mathbf{x}_t)$ by feeding an LSTM's output into two linear modules to estimate the mean $\mu_{\mathbf{z}_t}$ and co-variance $\Sigma_{\mathbf{z}_t}$ of the latent variable. Then, the randomly sampled $\mathbf{z}$ from the posterior $p(\mathbf{z}_t|\mathbf{x}_t)$ feeds into the decoder's LSTM. The final outputs are the reconstruction mean $\mu_{\mathbf{x}_t}$ and co-variance $\Sigma_{\mathbf{x}_t}$.

We apply a denoising autoencoding criterion [143] to the LSTM-VAE by introducing corrupted input with Gaussian noise, $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_{\text{noise}})$. We then replace the lower bound in Eq. (3.17) with a denoising variational lower bound $\mathcal{L}_{\text{dvae}}$ [165],

$$\begin{aligned}\mathcal{L}_{\text{dvae}} = &-D_{KL}(\tilde{q}_\phi(\mathbf{z}_t|\mathbf{x}_t)||p_\theta(\mathbf{z}_t)) \\ &+ \mathbb{E}_{\tilde{q}_\phi(\mathbf{z}_t|\mathbf{x}_t)}[\log p_\theta(\mathbf{x}_t|\mathbf{z}_t)],\end{aligned} \tag{3.18}$$

where $\tilde{q}_\phi(\mathbf{z}_t|\mathbf{x}_t)$ is an approximated posterior distribution given a corruption distribution

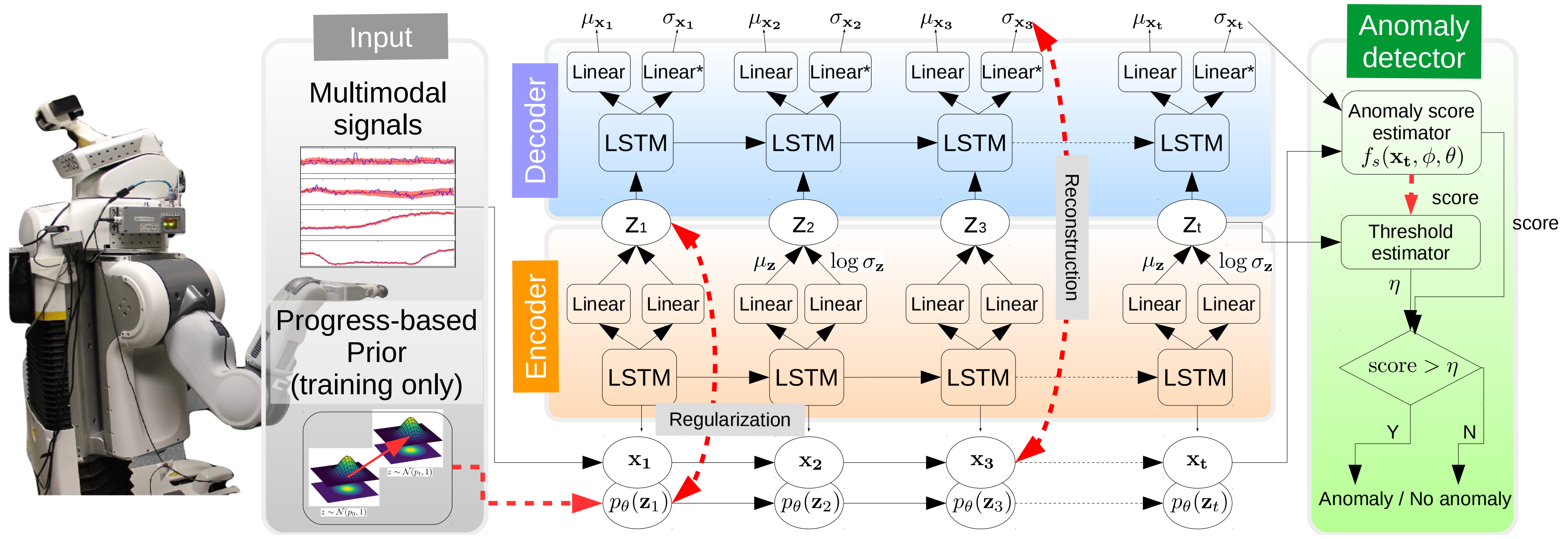

Input
Multimodal signals
Progress-based Prior (training only)
Decoder
Encoder
$\mu_{\mathbf{x_1}}$ $\sigma_{\mathbf{x_1}}$ $\mu_{\mathbf{x_2}}$ $\sigma_{\mathbf{x_2}}$ $\mu_{\mathbf{x_3}}$ $\sigma_{\mathbf{x_3}}$ $\mu_{\mathbf{x_t}}$ $\sigma_{\mathbf{x_t}}$
Linear
Linear*
LSTM
z_1 z_2 z_3 z_t
$\mu_{\mathbf{z}}$ $\log \sigma_{\mathbf{z}}$
Reconstruction
Regularization
$\mathbf{x_1}$ $\mathbf{x_2}$ $\mathbf{x_3}$ $\mathbf{x_t}$
$p_\theta(\mathbf{z}_1)$ $p_\theta(\mathbf{z}_2)$ $p_\theta(\mathbf{z}_3)$ $p_\theta(\mathbf{z}_t)$
Anomaly detector
Anomaly score estimator $f_s(\mathbf{x_t}, \phi, \theta)$
score
Threshold estimator
η
score $> \eta$
Y
N
Anomaly / No anomaly

Figure 3.18: Illustration of a multimodal anomaly detector with an unrolled LSTM-VAE model. The detector inputs multimodal signals to the model that compresses and reconstructs the inputs at each time step. The detector then determines an anomaly when a reconstruction-based anomaly score is over an estimated threshold η. In training, the model optimizes its parameters to have maximum regularization and minimum reconstruction error described in Eq. (3.17). The detector also trains a data-driven estimator that varies η with respect to state $\mathbf{z}$. Note that Linear* and LSTM layers have tanh and softplus activations, respectively. The red dash arrows are used for training only.

around $\mathbf{x}_t$. Given Gaussian distributions for $p(\tilde{\mathbf{x}}|\mathbf{x})$ and $q_\phi(\mathbf{z}|\mathbf{x})$, $\tilde{q}_\phi(\mathbf{z}_t|\mathbf{x}_t)$ can be represented as a mixture of Gaussians. For computational convenience, we use a single Gaussian, $\tilde{q}_\phi(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z}|\tilde{\mathbf{x}})$.

We introduce a progress-based prior $p(\mathbf{z}_t)$. Unlike conventional static priors using a normal distribution $\mathcal{N}(0,1)$, we vary the center of a normal distribution as $\mathcal{N}(\mu_p, \Sigma_p)$, where μ_p and Σ_p are the center and co-variance of the underlying distribution of multimodal inputs, respectively (see Fig. 3.19). This varying prior introduces the temporal dependency of time-series data into its underlying distribution by minimizing the difference between the approximated posterior and the prior. Unlike the RNN prior of Solch et al. [85] and the transition prior of Karl et al. [166], we gradually change μ_p from p_1 to p_T as the task execution progresses. In addition, the reconstruction performance and regularization loss depend on the distribution of a selected prior. We use an isotropic normal distribution where $\Sigma_p = I$ to simplify the prior and reduce hyper parameters. Note that we have tested various priors by changing its covariance matrix, but there was no noticeable difference. We can rewrite the regularization term of $\mathcal{L}_{\text{dvae}}$ as

$$\begin{aligned} & D_{KL}(\tilde{q}_\phi(\mathbf{z}_t|\mathbf{x}_t)||p_\theta(\mathbf{z}_t)) \\ & \approx D_{KL}(\mathcal{N}(\mu_{\mathbf{z}_t}, \Sigma_{\mathbf{z}_t})||\mathcal{N}(\mu_p, 1)). \\ & = \frac{1}{2}\left(\text{tr}(\Sigma_{\mathbf{z}_t}) + (\mu_p - \mu_{\mathbf{z}_t})^T(\mu_p - \mu_{\mathbf{z}_t}) - D - \log|\Sigma_{\mathbf{z}_t}|\right). \end{aligned} \tag{3.19}$$

To represent the distribution of high-dimensional continuous data, we use a multivariate Gaussian with a diagonal co-variance matrix. We can derive the reconstruction term in $\mathcal{L}_{\text{dvae}}$ as

$$\begin{aligned} & \mathbb{E}_{\tilde{q}_\phi(\mathbf{z}_t|\mathbf{x}_t)}[\log p_\theta(\mathbf{x}_t|\mathbf{z}_t)] \\ & = -\frac{1}{2}(\log(|\Sigma_{\mathbf{x}_t}|) + (\mathbf{x}_t - \mu_{\mathbf{x}_t})^T\Sigma_{\mathbf{x}_t}^{-1}(\mathbf{x}_t - \mu_{\mathbf{x}_t}) \\ & + D\log(2\pi)) \end{aligned} \tag{3.20}$$

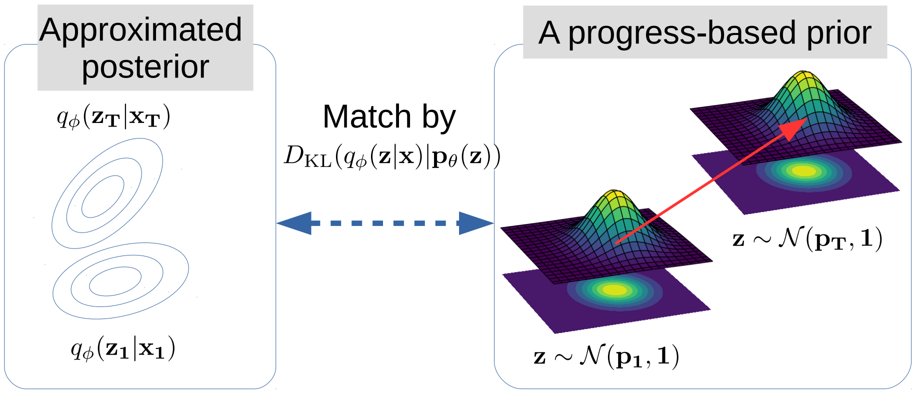

Approximated posterior
$q_\phi(\mathbf{z_T}|\mathbf{x_T})$
$q_\phi(\mathbf{z_1}|\mathbf{x_1})$
Match by
$D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})|\mathbf{p}_\theta(\mathbf{z}))$
A progress-based prior
$\mathbf{z} \sim \mathcal{N}(\mathbf{p_T}, \mathbf{1})$
$\mathbf{z} \sim \mathcal{N}(\mathbf{p_1}, \mathbf{1})$

Figure 3.19: Illustration of the progress-based prior. The center of the prior linearly changes from p_1 as initial progress to p_T as final progress.

We implemented the LSTM-VAE using stateful-LSTM models in the Keras deep learning library [167]. We trained the LSTM-VAE using an Adam optimizer with 3-dimensional latent variables and a 0.001 learning rate. We also use LSTM layers with *tanh*. Note that we are not using a sliding window in this work, but a window could be applied.

3.3.3 Anomaly Detection

We now introduce an online anomaly detection framework for multimodal sensory signals with state-based thresholding.

Anomaly Score

Our method detects an anomalous execution when the current anomaly score of an observation $\mathbf{x}_t$ is higher than a score threshold η,

$$\begin{cases} \mathrm{anomaly}, & \text{if } f_s(\mathbf{x}_t, \phi, \theta) > \eta \\ \neg\mathrm{anomaly}, & \text{otherwise}, \end{cases} \tag{3.21}$$

where $f_s(\mathbf{x}_t, \phi, \theta)$ is an anomaly score estimator. We define the score as the negative log-likelihood of an observation with respect to the reconstructed distribution of the observation through an encoding-decoding model,

$$f_s(\mathbf{x}_t, \phi, \theta) = -\log p(\mathbf{x}_t; \mu_{\mathbf{x}_t}, \Sigma_{\mathbf{x}_t}), \tag{3.22}$$

where $\mu_{\mathbf{x}_t}$ and $\Sigma_{\mathbf{x}_t}$ are the mean and co-variance of the reconstructed distribution, $\mathcal{N}(\mu_{\mathbf{x}_t}, \Sigma_{\mathbf{x}_t})$, from an LSTM-VAE with parameters ϕ and θ. A high score indicates an input has not been reconstructed well by the the LSTM-VAE. In other words, the input has deviated greatly from the non-anomalous training data.

State-based Thresholding

We introduce a varying threshold that changes over the estimated state of a task execution motivated by the dynamic threshold [42]. Depending on the state of task executions, reconstruction quality may vary. In other words, anomaly scores in non-anomalous task executions can be high in certain states, so varying the anomaly score can reduce false alarms and improve sensitivity. In this paper, the state is the latent space representation of observations. Given a sequence of observations, the encoder of LSTM-VAE is able to compute a state at each time step. By mapping states $\mathbf{Z}$ and corresponding anomaly scores S from a non-anomalous dataset, our method is able to train an expected anomaly score estimator $\hat{f}_s : \mathbf{z} \to s$. We use support vector regression (SVR) to map from a multidimensional input $\mathbf{z} \in \mathbf{Z}$ to a scaler s using a radial basis function (RBF) kernel. To control sensitivity, we add a constant c into the expected score and represent the state-based threshold as $\eta = \hat{f}_s(\mathbf{z}) + c$.

Training and Testing Framework

Algorithm 2 shows the training framework of our LSTM-VAE-based anomaly detector. Given a set of non-anomalous training and validation data, $(\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}})$, the framework aims to output the optimized parameters (ϕ, θ) of an LSTM-VAE and an expected anomaly score estimator $\hat{f}_s$. Note that we represent N sequences of multimodal observations as

$\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}$. N_{train} and N_{val} denote the numbers of training and validation data, respectively. We also represent the encoder and decoder functions as $f_\phi : \mathbf{x}_t \rightarrow \mathbf{z}_t$ and $g_\theta : \mathbf{z}_t \rightarrow (\mu_{\mathbf{x}_t}, \Sigma_{\mathbf{x}_t})$, respectively. Then, we denote the function of the serially connected encoder and decoder (i.e., autoencoder) by $f_{\phi,\theta}$ with noise injection.

The framework pre-processes $\mathbf{X}_{\text{train}}$ and $\mathbf{X}_{\text{val}}$ by resampling those to have length T and normalizing their individual modalities in the range of $[0, 1]$ with respect to $\mathbf{X}_{\text{train}}$. The framework then starts to train the LSTM-VAE with respect to $\mathbf{X}_{\text{train}}$ maximizing $\mathcal{L}_{\text{dvae}}$ and stops the training when $\mathcal{L}_{\text{dvae}}$ does not increase for 4 epochs. Then it extracts a set of latent space representations and corresponding anomaly scores from $\mathbf{X}_{\text{val}}$ as the training set for $\hat{f}_s$. Finally, this framework returns the trained SVR object as well as the LSTM-VAE's parameters. Note that we reset the state of the LSTM in the beginning of a sequence of data only.

In testing, the detector aims to detect an anomaly in real time. Algorithm 3 shows the pseudo code for the online detection process. In each loop, the detector takes multimodal input $\mathbf{x}$ and scales its individual dimension with respect to the scaled $\mathbf{X}_{\text{train}}$. The detector then estimates a latent variable and the parameters of the expected distribution. When the anomaly score of the current input is higher than η, our detector determines the current task execution is anomalous and returns the decision. We control the sensitivity of the detector by adjusting c.

3.3.4 Experimental Setup

Instrumental Setup

Our system uses a PR2 from Willow Garage, a general-purpose mobile manipulator with two 7-DOF arms and powered grippers. To prevent possible hazards, we used a low-level PID controller with low gains and a $50\,\text{Hz}$ mid-level model predictive controller from [156] without haptic feedback. We used the following sensors: an RGB-D camera with a microphone (Intel SR300) on the right wrist, a force/torque sensor (ATI Nano25) on the utensil

Algorithm 2: Training algorithm for an LSTM-VAE-based anomaly detector

```
   input  : $\mathbf{X}_{\text{train}} \in \mathbb{R}^{N_{\text{train}} \times T \times D}$, $\mathbf{X}_{\text{val}} \in \mathbb{R}^{N_{\text{val}} \times T \times D}$
   output: $\phi$, $\theta$, $f_\eta$
 1 $\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}$ = Preprocessing($\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}$) ;
 2 $\phi, \theta \leftarrow$ train LSTM-VAE with $(\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}})$;
 3 $\mathbf{Z} = \emptyset, \mathbf{S} = \emptyset$ ;
 4 for $i \leftarrow 1$ to $N_{\text{val}}$ do
 5     Reset the state of LSTM-VAE;
 6     for $j \leftarrow 1$ to $T$ do
 7         $\mathbf{z} \leftarrow f_\phi(\mathbf{X}_{\text{val}}(i, j))$;
 8         $\mu_x, \sigma_{\mathbf{x}} \leftarrow f_{\phi,\theta}(\mathbf{X}_{\text{val}}(i, j))$;
 9         $s \leftarrow f_s(\mathbf{x}_{\text{val}}(i, j), \mu_{\mathbf{x}}, \sigma_{\mathbf{x}})$;
10         Add $\mathbf{z}$ and $s$ into $\mathbf{Z}$ and $\mathbf{S}$, respectively.
11     end
12 end
13 $\hat{f}_s \leftarrow$ train an SVR with $(\mathbf{Z}, \mathbf{S})$.
```

Algorithm 3: Testing algorithm for an LSTM-VAE-based anomaly detector.

```
   input  : $\mathbf{x} \in \mathbb{R}^D$
   output: Anomaly or ¬Anomaly
 1 while *True* do
 2     $\mathbf{x} \leftarrow$ get current multimodal data;
 3     $\mathbf{x} \leftarrow Preprocessing(\mathbf{x})$;
 4     $\mathbf{z} \leftarrow f_\phi(\mathbf{x})$;
 5     $\mu_{\mathbf{x}}, \Sigma_{\mathbf{x}} \leftarrow f_{\phi,\theta}(\mathbf{x})$;
 6     if $f_s(\mathbf{x}; \mu_{\mathbf{x}}, \Sigma_{\mathbf{x}}) > \hat{f}_s(\mathbf{z}) + c$ then
 7         return Anomaly ;
 8     end
 9 end
```

handle, joint encoders, and current sensors. These sensors measure mouth position and sound, force on the utensil, spoon position, and joint torque, respectively.

Using a web-based graphical user interface, the user can send three commands (i.e., *scooping/stabbing*, *feeding*, and *clean spoon*) to the robot. In a typical execution, the user will send a *scooping/stabbing* command followed by a *feeding* command. The robot scoops or stabs food from a bowl given *scooping/stabbing*, and then brings the food into the user's mouth location estimated using the RGB-D camera given *feeding*. The user can send *clean spoon* so that the robot can drag the spoon across a bar. The robot uses pre-defined motions which adapts to the configuration of the user and robot.

Data Collection

We used data from 1,555 feeding executions collected from 24 able-bodied participants. 16 participants were male and 8 were female, and the age range was 19-35. We conducted the studies with approval from the Georgia Tech Institutional Review Board (IRB).

We divided our data into two subsets: a training/testing dataset collected from our previous work [163] and a pre-training dataset. The training/testing dataset consists of data from 352 executions (160 anomalous and 192 non-anomalous) collected from 8 participants who used the feeding system with yogurt and a silicone spoon. The pre-training dataset uses data from 1,203 non-anomalous executions from 16 newly recruited participants who used various foods and utensils. The broader range of the dataset allowed us to initialize the weights of the LSTM-VAE and reduce the impact of overfitting in fine tuning. Among the dataset, 559 non-anomalous executions were from 9 participants who used 3 types of food and corresponding utensils: cottage cheese and silicone spoon, watermelon chunks and metal fork, and fruit mix and plastic spoon. An experimenter also conducted 428 non-anomalous executions as a self-study with 6 foods (yogurt, rice, fruit mix, watermelon chunks, cereal, and cottage cheese) and 5 utensils (small/large plastic spoons, a silicone spoon, and plastic/metal forks) (see Figure 3.20). We also collected additional data from

216 non-anomalous executions from 6 participants who used yogurt and a silicone spoon.

Experimental Procedure

Each participant performed anomalous and non-anomalous feeding executions while the participant, experimenters, or the system produced anomalies. We randomly determined the order of these executions. In order to approximate one form of limited mobility that people with disabilities may have, we instructed the participants to not move their upper bodies and to eat food off the utensil using their lips. We defined 12 types of representative anomalies through fault tree analysis [17]: touch by user, aggressive eating, utensil collision by user, sound from user, face occlusion, utensil miss by user, unreachable location, environmental collision, environmental noise, utensil miss by system fault, utensil collision by system fault, and system freeze (see Fig. 3.21). For anomalies caused by the user, we instructed the participants through demonstration videos and verbal explanation. The participant controlled the details of their actions such as timing and magnitude.

Pre-processing

For each feeding execution, we collected 17 sensory signals from 5 sensors: *sound energy* (1), *force* (3) applied on the end effector, *joint torque* (7), *spoon position* (3), and *mouth position* (3), where the number in parentheses represents the dimension of signals. We zeroed the initial value and resampled each signal to have $20\,\mathrm{Hz}$ for the robot's actual anomaly check frequency. We then scaled signals in the non-anomalous dataset to have a value between 0 and 1. Corresponding to this scale, we also scaled signals from the anomalous dataset. Finally, we have a sequence of tuples per execution (i.e., *sequence length* $\times$ 17). Note that the executions have timing variations due to the variability of the robot's posture, each participant's seating, and human actions during the feeding.

For visualization and comparison purposes, we also extracted 4-dimensional hand-engineered features used in our previous work [142]: *sound energy*, *1st joint torque*, *ac-*

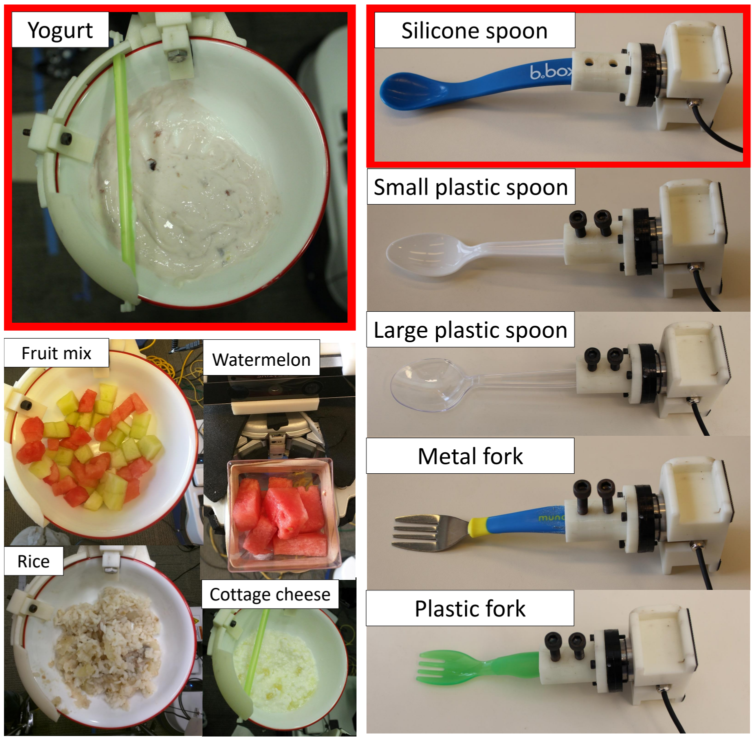


Figure 3.20: **Left:** Examples of food used in our experiments. **Right:** The 3D-printed utensil handle and 5 utensils used. Red boxes show yogurt and silicone spoon used for our training/testing dataset.

cumulated force, and *spoon-mouth distance*. Here, we used *sound energy*[5] instead of raw $44\,100\,\mathrm{kHz}$ 16 bit PCM encoding since the under sampling could miss auditory anomalies.

Baseline Methods

To evaluate the performance of the proposed method, we implemented 5-baseline methods,

- RANDOM: A random binary classifier in which we control its sensitivity by weighting a class.
- OSVM: A one-class SVM-based detector trained with only non-anomalous executions. We move a sliding window (of size 3 in time like EncDec-AD [80]) one step at a time. We control its sensitivity by adjusting the number of support vectors.

[5]Root mean square (RMS) of 1,024 frames.

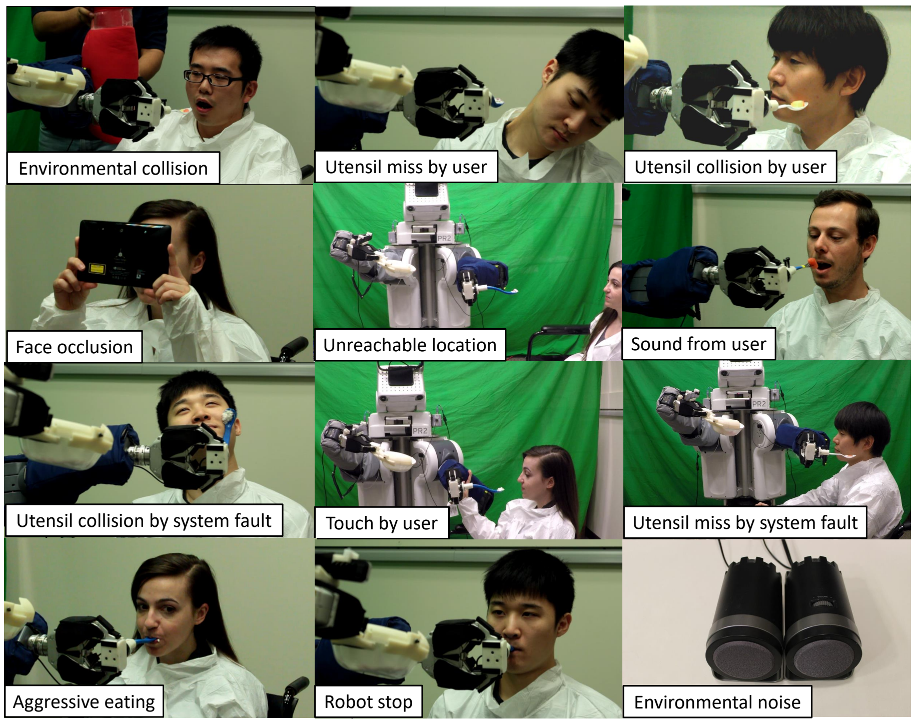


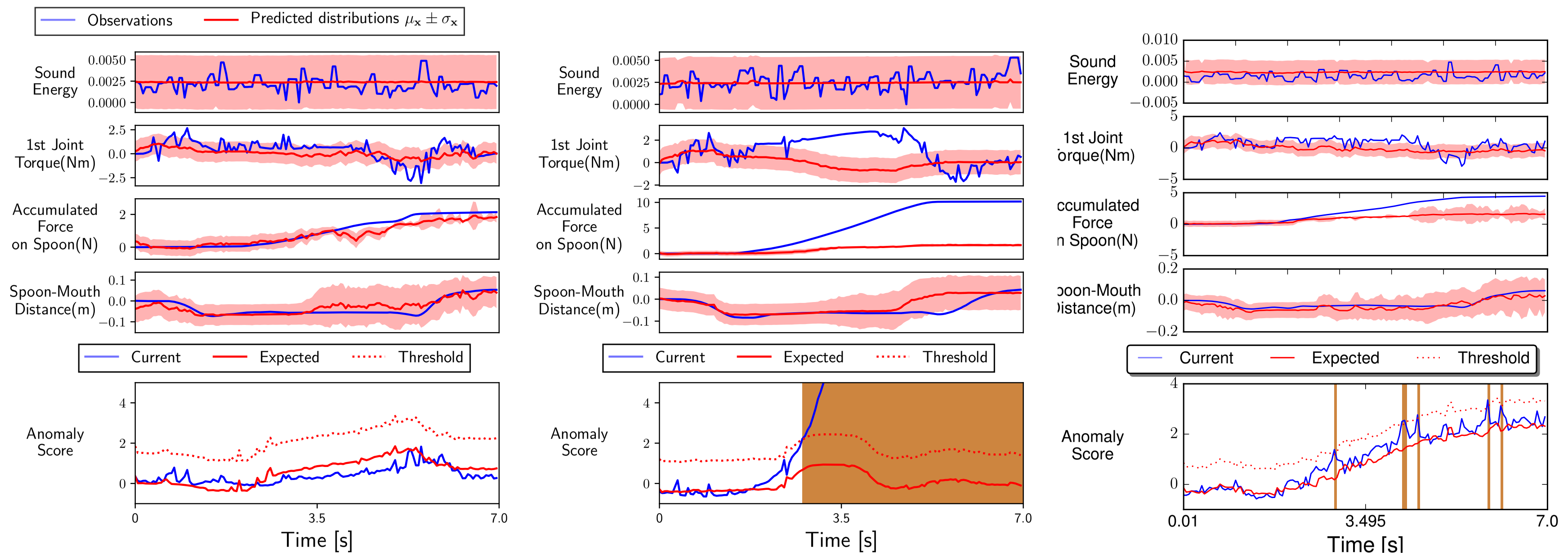
Figure 3.21: 12 representative anomalies caused by the user, the environment, or the system in our experiments.

- HMM-GP: A likelihood-based classifier using an HMM introduced in [142]. We vary the likelihood threshold with respect to the distribution of hidden states.

- AE: A reconstruction-based anomaly detector using a conventional autoencoder with a 3 time-step sliding window based on [168].

- EncDec-AD: A reconstruction-based anomaly detector using an LSTM-based autoencoder [80]. We use window size $L = 3$ as in the paper, but unlike the paper we use a diagonal co-variance matrix when we model the distribution of reconstruction-error vectors.

From now on, we will also use the term LSTM-VAE to refer to our LSTM-VAE-based detector.



(a) A non-anomalous execution (b) An anomalous execution (c) An anomalous execution with small contacts

Figure 3.22: Visualization of the reconstruction performance and anomaly scores over time using an LSTM-VAE. The upper four sub graphs show observations and reconstructed observations' distribution. The lower sub graphs show current and expected anomaly scores. The dashed curve shows a state-based threshold where the LSTM-VAE determines an anomaly when current anomaly score is over the threshold. Brown vertical lines represent the time of anomaly detection.

3.3.5 Evaluation and Discussion

We first investigated the reconstruction function of the LSTM-VAE. The upper 4 sub graphs in Fig. 3.22 show the expected distribution of 4 hand-engineered features from non-anomalous and anomalous feeding executions in the robot-assisted feeding task. For Fig. 3.22a, the observed features (blue curves) and the mean of expected distribution (red curves) show a similar pattern of change over time. On the other hand, in anomalous executions (see Fig. 3.22b and Fig. 3.22c), the LSTM-VAE resulted in large deviations between observed and reconstructed *accumulated force* since the pattern by the collision is not easily observable from non-anomalous executions. Consequently, we can observe the anomaly score (blue curve) gradually increases after the onset of the deviation from the lower sub graphs of Figure 3.22b. Note that the anomalous executions came from large and small face-spoon collisions caused intentionally by the user. The sound energy graphs show environmental noise only.

The anomaly score metric is effective in distinguishing anomalies. Fig. 3.23 shows the distributions of the anomaly scores over time of a participant's 24 anomalous and 20 non-anomalous feeding executions during leave-one-person-out cross validation. The blue and red shaded regions show the mean and standard deviation of non-anomalous and anomalous executions' anomaly scores, respectively. The score of non-anomalous executions shows a specific pattern of change with a smaller average and variance than that of anomalous executions, making anomalies easily distinguishable from non-anomalies.

The lower sub graphs of Fig. 3.22 also show the state-based threshold is capable of achieving a tighter anomaly decision boundary (red dash lines) than a fixed threshold over time. The expected anomaly scores (red curves) and the actual scores (blue curves) show a similar pattern of change. However, the expected score is lower than the actual score given an anomaly. Brown vertical lines show the time of anomaly detection where the first detection time matches with the initial increase of accumulated force.

We compared our LSTM-VAE with 5 other baseline methods through a leave-one-

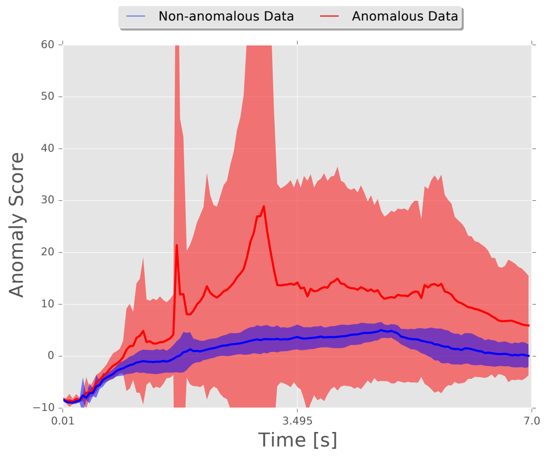


Figure 3.23: An example distributions of anomaly scores from a participant's 20 non-anomalous and 24 anomalous executions over time.

person-out cross-validation method (see Table 3.4). Given the training/testing dataset, we used data from 7 participants for training and tested with the data from the remaining 1 participant. In this evaluation, we pre-trained each method using the pre-training dataset in addition to the dataset from the 7 participants. We then fine-tuned each method with the data from 7 participants. Note that we only trained the OSVM with the pre-training dataset and we did not succeed in training HMM-GP due to underflow errors caused by high-dimensional input.

Our method outperformed the other methods with 0.044 higher AUC than the next best method, HMM-GP, when using 4 hand-engineered features. When using 17 sensory signals with the additional pre-training dataset, our method resulted in the highest performance of AUC. The time-series autoencoding methods, EncDec-AD and LSTM-VAE, improved AUC but the others did not when we increased inputs. This indicates the autoencoding is capable of extracting effective information from the high-dimensional signals without significant feature engineering effort. In addition, we tested with double the window size to investigate its effect but OSVM and AE resulted in only small improvements.

Fig. 3.24 shows ROC curve changes given two thresholding techniques: fixed and state-based thresholds. To investigate the influence of VAE, we implemented an LSTM-based encoder-decoder (LSTM-AE) with the two techniques by excluding VI. The LSTM-

VAE with state-based thresholding outperformed that with conventional fixed thresholding, resulting in higher true positive rates given the same false positive rates. The LSTM-AE with both thresholding techniques resulted in lower true positive rates than the LSTM-VAE. Particularly, the LSTM-AE with fixed thresholding shows the lowest performance. These results indicate the VAE is helpful in reconstructing the multimodal time-series data. The results also show the VAE provides better state distribution over time for threshold regression than the vanilla autoencoder. In this evaluation, we used 17 sensory signals with the pre-training dataset.

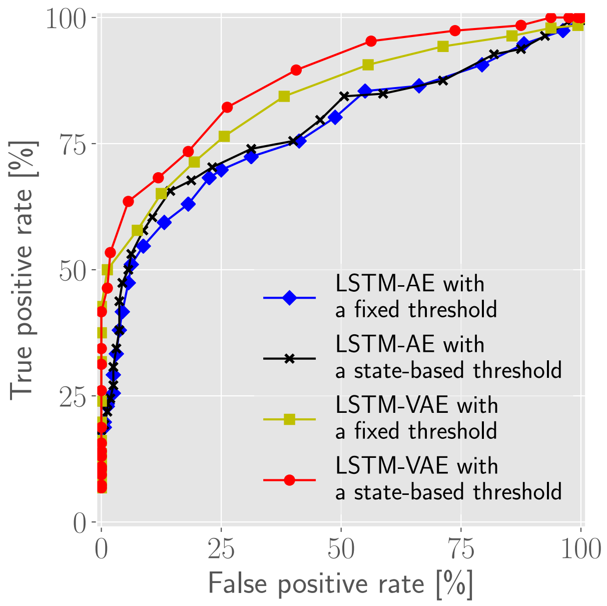


Figure 3.24: Receiver operating characteristic (ROC) curves to compare the performance of LSTM-VAEs with and without a state-based threshold.

Table 3.4: Comparison of the LSTM-VAE and 5 baseline methods with two types of inputs. Numbers represent the area under the ROC curve (AUC). s represents the length of a window.

Input	Random	OSVM		HMM-GP	AE		EncDec-AD		LSTM-VAE
		$s=3$	$s=6$		$s=3$	$s=6$	$s=3$	$s=6$	
4 hand-engineered features	0.5121	0.7427	-	0.8121	0.8123	-	0.7995	-	**0.8564**
17 raw sensory signals	0.5052	0.7376	0.7408	N/A	0.8012	0.8108	0.8075	0.8021	**0.8710**

CHAPTER 4
MULTIMODAL ANOMALY CLASSIFICATION

4.1 Overview

In this chapter, we focus on the problem of classifying and responding to common anomalies, particularly during the robot-assisted feeding.

While greater autonomy and general-purpose robots have the potential to improve assistance, they also increase the complexity of robotic systems. This increase in complexity can lead to a higher likelihood of anomalies when providing assistance in the real world. While physically assisting a person with disabilities, anomalies could decrease system safety, effectiveness, and usability. Previously, we presented a monitoring system that uses multimodal sensing to detect an anomaly in Chapter 3. However, the monitoring system does not provides detailed information, such as the type and cause of an anomaly, which can be used for adapting and improving robot-assistant systems. It can also be used to inform users of the robot's detailed status. Below, we will introduce a classification framework into the monitoring system so the assistant system is able to select adequate recovery plan depending on the significance of detected anomalies.

Our execution monitor consists of a data-driven anomaly detector and classifier (see Figure 4.1). For the detector, we combine two multimodal anomaly detectors, referred to as HMM-D [42], that use multivariate hidden Markov models (HMMs) and dynamic thresholds to determine anomalies. Our anomaly classifier uses a multilayer perceptron (MLP) with selected input features extracted from HMMs, raw sensory signals, and a convolutional neural network (CNN). To use unexpected changes of signals, our system extracts conditional probabilities of each signal over others and then extract its temporal changes using temporal pyramid pooling as addressed in [169]. We also extract bottleneck features, the output of the CNN given an image. After detection, an MLP fuses these features and estimates the most probable class of the anomaly among 12 classes selected through fault tree analysis (see Figure 4.2).

We evaluated the detection and identification performance of our execution monitor using a robot-assisted feeding dataset where a PR2 fed 8 able-bodied participants. We then evaluated our execution monitoring system with Henry Evans, a person with quadriplegia in California, USA. The robot recorded haptic, auditory, kinematic, and visual data from a variety of sensors during the feeding task. Our execution monitor successfully detected and identified a variety of anomalous executions, such as the spoon missing the person's mouth, unexpected collisions during feeding, and a loud utterance from the care receiver or a nearby caregiver. Our method resulted in substantially higher classification accuracy when compared against six other baseline classification methods from the literature.

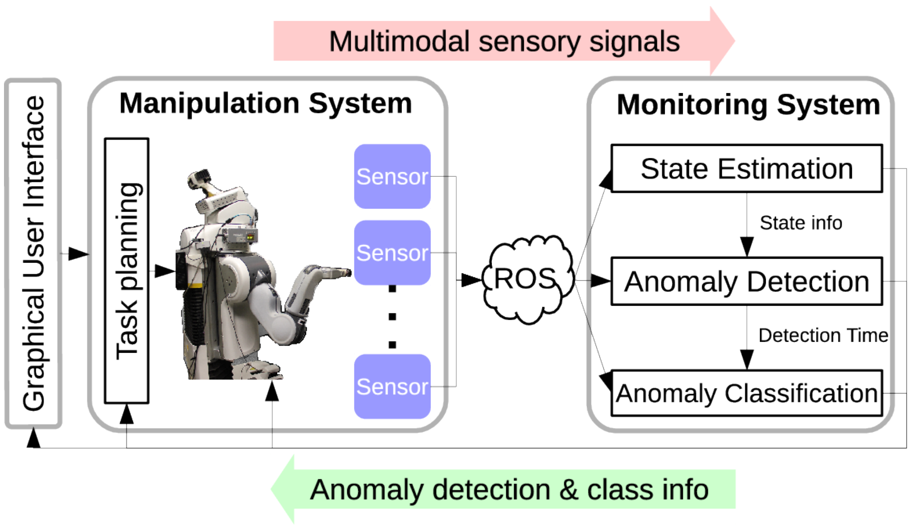


Figure 4.1: Overview of the multimodal execution monitor.

4.2 Anomaly Classes

According to Ogorodnikova's work [17], we classify the causes of anomalies into 3 groups: engineering, human, and environmental conditions. The engineering condition includes errors from mechanical failure (e.g., loose connection), controller failure (e.g., uncontrolled

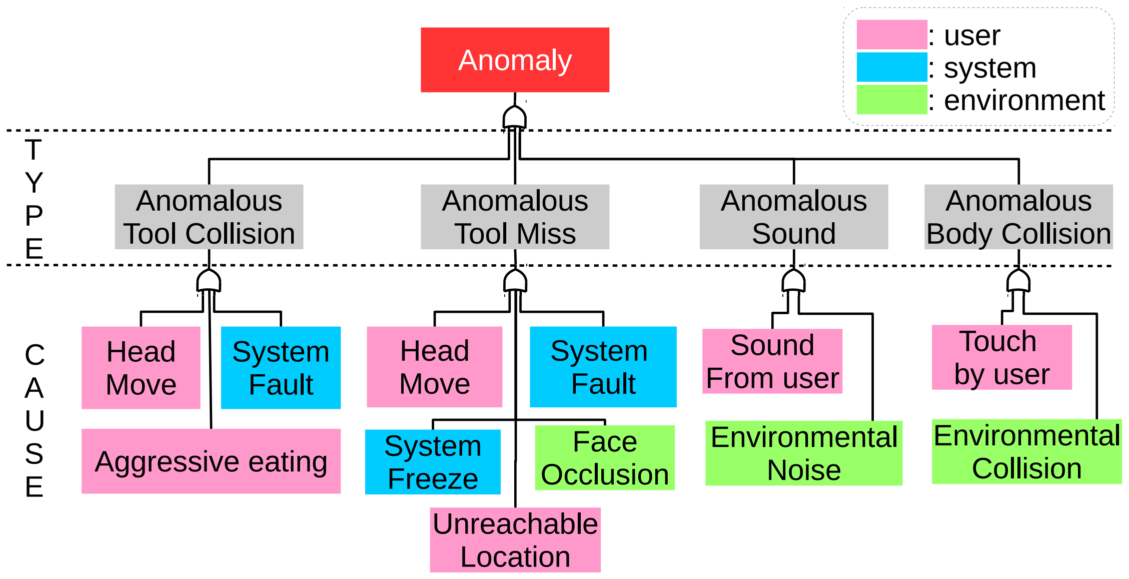


Figure 4.2: Fault tree analysis for the robot-assisted feeding task.

speed), and programming bugs. The human condition includes inadequate teaching of robot path or incorrect decision of initial conditions. The environmental condition includes poor sensing abilities, light condition, etc. Although robotic assistance might benefit from robots that are able to identify the cause of an anomaly, causal inference is difficult due to the complexity of real-world manipulation. Instead, we focus on the detection and classification of representative anomalies that are more likely to occur during the direct task at hand. For robot-assisted feeding, we identified 12 anomalies through fault tree analysis which is a deductive analysis approach for resolving system hazards into their causes. Figure 4.2 illustrates the results of our fault tree analysis. In the figure, depth 1 and 2 represent the types and causes of anomalies, respectively. The three colors used in depth 2 represent the causal groups from [17, 18].

subsectionAnomaly Detection Our system detects anomalies with HMM-D detectors, which we introduced in [42]. HMM-D is a binary (one-class) detector that learns a model from non-anomalous task executions and detects anomalies when the log-likelihood of a sequence of input signals is lower than a time-varying threshold. HMM-D dynamically changes the threshold depending on the progress of a current task execution. In this study, we used 25 hidden states for HMMs and 25 Gaussian radial basis functions for the threshold selection.

In contrast to our previous work, our new system uses two HMM-D anomaly detectors, each responsible for modeling a different set of sensory features. If either detector detects an anomaly, then the system detects an anomaly. In practice, we found that decomposing anomaly detection in this way resulted in improved system performance. The two detectors use the following sensory features:

- 1st Detector: *sound energy*, *1st joint torque*, *accumulated force*, and *spoon-mouth distance*

- 2nd Detector: *spoon speed*, *force*, *desired spoon displacement*, and *spoon-mouth distance*.

Before training each of the two HMM-D detectors, we first extract hand-engineered features from raw sensory signals after resampling and scaling, as described in [42]. Each detector uses four hand-engineered features that we found resulted in improved detection performance[1] based on cross-validation tests (see Section 4.4.4).

We trained each HMM-D detector using the Baum-Welch algorithm as defined in the General Hidden Markov Model library (GHMM) (http://www.ghmm.org/). This training results in a set of HMM parameters λ which includes a transition probability matrix $\mathbf{A} \in \mathbb{R}^{\mathbf{25 \times 25}}$ and Gaussian emission probabilities B. We then found parameters for the dynamically-changing threshold, which is defined by the expected log-likelihood $\hat{\mu}$ and its standard deviation $\hat{\sigma}$.

4.3 Anomaly Classification

We introduce a supervised data-driven classifier that identifies representative anomalies from multimodal features.

[1] Area under curve (AUC) for a receiver operating characteristic curve

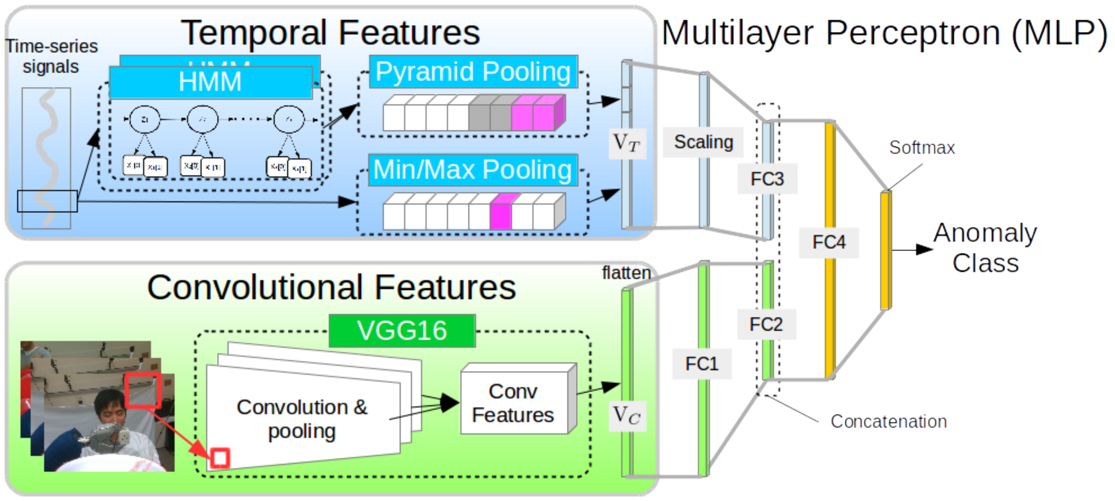


Figure 4.3: Illustration of our anomaly classification network. The MLP outputs the most probable class of an anomaly using temporal and convolutional features. The size of the FC1-FC4 layers are 1024, 128, 128, and 256, respectively.

4.3.1 Feature Extraction

Our monitoring system extracts two groups of multimodal features, temporal and convolutional features (see Figure 4.3).

Temporal Features

In the previous Chapter 3, we found that particular types of anomalies tended to be more apparent in a subset of modalities. Consequently, the extent to which the feature from particular modalities is unexpected could be useful when classifying anomalies. It could be also helpful to determine the largest anomaly given multiple anomalies. To measure the unexpected change in a feature, our system estimates the likelihood of each feature with respect to the other features given by the HMMs used in anomaly detection. We refer to this estimate as a conditional likelihood. The conditional likelihood of a feature X_i is

$$P(X_i|X_{\mathbb{S}\setminus\{i\}}, \lambda_{\mathbb{S}}) = \frac{P(X_{\mathbb{S}}|\lambda_{\mathbb{S}})}{P(X_{\mathbb{S}\setminus\{i\}}|\lambda_{\mathbb{S}})} \tag{4.1}$$

$$= \frac{P(X_{\mathbb{S}}|\lambda_{\mathbb{S}})}{P(X_{\mathbb{S}\setminus\{i\}}|\lambda_{\mathbb{S}\setminus\{i\}})}, \tag{4.2}$$

where $X_{\mathbb{S}}$ is a set of features in the HMM, $\lambda_{\mathbb{S}}$ is the HMM's parameters, and $\lambda_{\mathbb{S}\setminus\{i\}}$ is a part of λ that excludes the elements related to X_i. We prove the equivalence of the denominators in Equation (4.1) and (4.2) as follows

Lemma 4.3.1. $P(X_{\mathbb{S}\setminus\{i\}}|\lambda_{\mathbb{S}})$ equals $P(X_{\mathbb{S}\setminus\{i\}}|\lambda_{\mathbb{S}\setminus\{i\}})$

Proof. Let $X_{\mathbb{S}}$ be a set of all available time-series features. Given a single-state HMM, its parameter set $\lambda_{\mathbb{S}}$ represents output mean $\mu_{\mathbb{S}}$ and co-variance $\Sigma_{\mathbb{S}}$. Now, we show two marginalization processes for ith feature in $X_{\mathbb{S}} \sim \mathcal{N}(\mu_{\mathbb{S}}, \Sigma_{\mathbb{S}})$.

First, the marginalization of the likelihood $P(X_{\mathbb{S}}|\mu_{\mathbb{S}}, \Sigma_{\mathbb{S}})$ over the ith feature using integration is

$$\sum_{X_i} P(X_{\mathbb{S}\setminus\{i\}}, X_i|\mu_{\mathbb{S}}, \Sigma_{\mathbb{S}}) \tag{4.3}$$

$$= P(X_{\mathbb{S}\setminus\{i\}}|\mu_{\mathbb{S}}, \Sigma_{\mathbb{S}}). \tag{4.4}$$

Second, the marginal distribution of a Gaussian is also a Gaussian; given $X_{\mathbb{S}} \sim \mathcal{N}(\mu_{\mathbb{S}}, \Sigma_{\mathbb{S}})$, $P(X_{\mathbb{S}\setminus\{i\}}) = \mathcal{N}(\mu_{\mathbb{S}\setminus\{i\}}, \Sigma_{\mathbb{S}\setminus\{i\}})$ [170]. We can represent the marginal likelihood of $X_{\mathbb{S}}$ over ith feature as $P(X_{\mathbb{S}\setminus\{i\}}|\mu_{\mathbb{S}\setminus\{i\}}, \Sigma_{\mathbb{S}\setminus\{i\}})$.

Both marginalization results must be equal. Thus, $P(X_{\mathbb{S}\setminus\{i\}}|\lambda_{\mathbb{S}})$ equals $P(X_{\mathbb{S}\setminus\{i\}}|\lambda_{\mathbb{S}\setminus\{i\}})$. □

In this dissertation, we use multivariate Gaussian emissions in HMMs. The marginal distribution of a multivariate Gaussian distribution is a Gaussian: $P(x_1, ..., x_n) = \mathcal{N}(\mu_{1:n}, \mathbf{\Sigma}_{1:n})$ given $\mathbf{X} \sim \mathcal{N}(\mu, \Sigma)$ [170]. Thus, the denominator of Eq. (4.1) can be converted to the denominator of Eq. (4.2). For computational convenience, we use the logarithm of (4.2), i.e., $l_i = logP(X_i|X_{\mathbb{S}\setminus\{i\}}, \lambda_{\mathbb{S}})$. At each time step t, we extract a total of 8 conditional log-likelihoods from two 4-dimensional HMMs and concatenate these to create a feature vector $\mathbf{v}_t$ in $\mathbb{R}^8$,

$$\mathbf{v}_t = \{l_1^1, l_2^1, l_3^1, l_4^1, l_1^2, l_2^2, l_3^2, l_4^2\}, \tag{4.5}$$

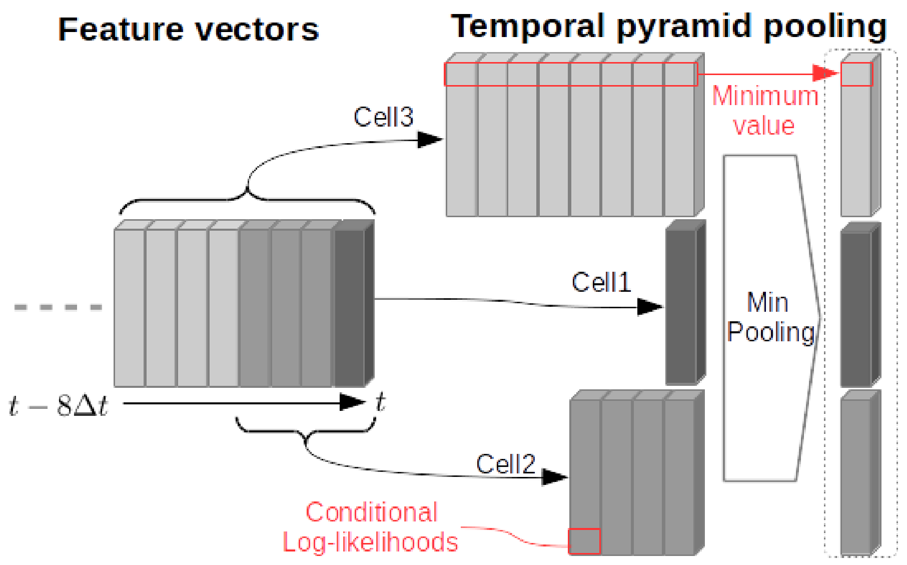


Figure 4.4: Illustration of the temporal pyramid pooling process. The boxes show the 3-level temporal partition of a sequence. The output feature vector is of size $8 \times 3 = 24$.

where the superscript shows the identity of the HMM.

To represent the temporal change of this feature vector, we perform 3 levels (1-4-8) of temporal pyramid pooling [169] given the last 8 time steps of feature vectors $\mathbf{v}_{[t-8\Delta t:t]}$. Figure 4.4 shows this pooling process for which we partition the feature vectors into 3 cells and pool the minimum value per feature (i.e., conditional log-likelihood of each feature). The pooling from 3 cells give 3 vectors, $\mathbf{v}^1$, $\mathbf{v}^4$, and $\mathbf{v}^8$. We then concatenate the vectors to form a single vector $[\mathbf{v}^1, \mathbf{v}^4, \mathbf{v}^8]$ in $\mathbb{R}^{24}$. For this dissertation, we chose minimum pooling since a drop in likelihood would indicate an unexpected change in the features.

In addition to the conditional log-likelihood features, we include 7 additional features that can be useful in identifying the cause of an anomaly. These include: *frontal sound amplitude*, *sound source direction (azimuth angle)*[2], *x-direction force*, *y-direction force*, *contact force on the whole-arm tactile sensing skin*, *distance between the robot's torso and the person's mouth*, and *spoon-mouth angular difference*. For each feature, we pool a minimal or maximal value over the last 20 time steps, which we refer to as min-max pooling (see Algorithm 4). This pooling enables us to extract the unexpected change of each feature

[2]We localize the source of sound using interaural time differences [171].

since the 7 features are usually static in non-anomalous executions. We empirically decided on 20 time steps for the pooled features, $\mathbf{v}^e$ in $\mathbb{R}^7$, to include only recent feature changes. The final feature vector $\mathbf{V}_T = [\mathbf{v}^1, \mathbf{v}^4, \mathbf{v}^8, \mathbf{v}^e]$ is of length 31 ($= 8 \times 3 + 7$).

Algorithm 4: Min-max Pooling

```
  Data: A sequence of observations, x
  Result: Pooled value
1 if $|min(\mathbf{x})| > |max(\mathbf{x})|$ then
2     return min(x) ;
3 else
4     return max(x) ;
5 end
```

Convolutional Features

The interpretation of visual information can help to better classify anomalies during feeding. Our algorithm extracts the output of a convolutional neural network (CNN) as bottleneck features [172], for images collected by the PR2. The features can represent the existence of objects around the work space. In this dissertation, we use the VGG16 CNN model which has been trained on the ImageNet dataset with 1,000 classes [173]. When an anomaly is detected, our network takes as input a 224x224 RGB image of the person captured from the wrist-mounted camera. Note that the PR2 always positions the camera in front of the person to ensure his or her face is in the captured image (see captured images in Figure 4.3). We then flatten output features ($\in \mathbb{R}^{512\times7\times7}$) to a vector $\mathbf{V}_C$ which is input into a multilayer perceptron (MLP) (see Figure 4.3). In this work, we used Keras, a deep learning library [167], with the Tensorflow back end to load the VGG16 network and extract bottleneck features.

4.3.2 Multilayer Perceptron (MLP) Classification

The execution monitor uses a multilayer perceptron (MLP) to classify the types and causes of anomalies given the multimodal temporal and convolutional features. An MLP is a feedforward artificial neural network with fully-connected layers. The right side of Figure 4.3 presents our MLP structure. Our MLP consists of three fully-connected layers with rectified linear units (ReLU). A softmax function is applied to the final layer for multiclass classification.

Before fusing the temporal and convolutional features, we feed the convolutional features through the first fully-connected layer of our MLP. We then concatenate the temporal features ($\in \mathbb{R}^{128}$) and the first layer's output ($\in \mathbb{R}^{128}$) to a vector ($\in \mathbb{R}^{256}$) similar to common CNN-LSTM models [174]. Figure 4.5 shows the distribution of the concatenated multimodal features using data from trials with 8 able-bodied participants. We display the first two principal components from a principal component analysis (PCA).

To train this network, we used only positive (anomalous) feeding trials. It is difficult to label exactly when an anomaly has started to occur and the anomaly detector's sensitivity can influence the timing of detections. To account for this, we first set the thresholds of the HMM-D detectors to maximize detection accuracy given a training dataset. We then collected the temporal and convolutional features when the system first detected an anomaly. We also collected additional feature sets over 10 time steps before and after the time at which the detection occurred to account for variation in the timing of anomaly detections. Before concatenating the features at each time step, we performed feature-wise scaling for the temporal features to have zero mean and unit variance. We trained the MLP classifier initialized with uniformly distributed random weights using a stochastic gradient descent (SGD) optimizer with RMSProp. We then fine tuned it with a conventional SGD optimizer. To avoid overfitting, we added dropout and L2-regularization to each layer [175].

During real time experiments, our system extracts features and performs anomaly classification only after an anomaly has been detected.

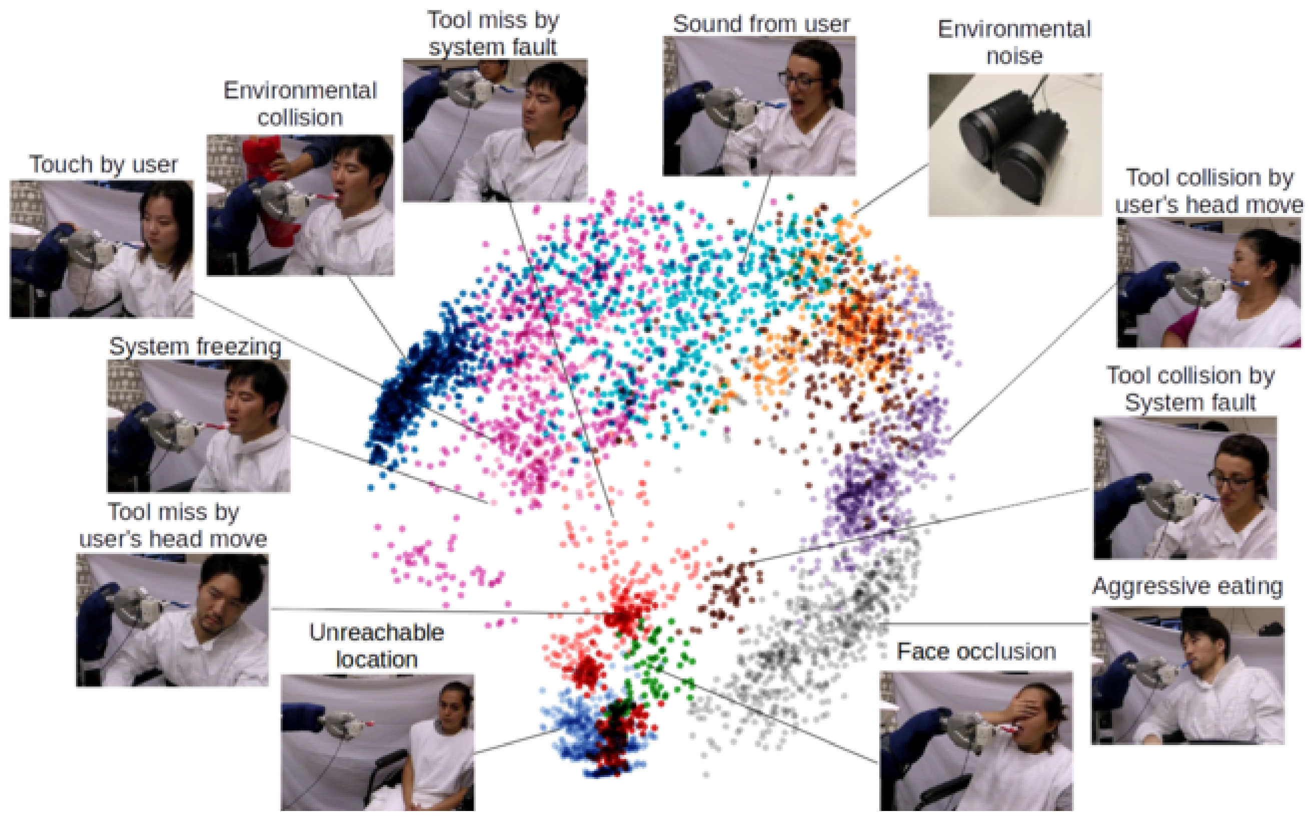


Figure 4.5: Distribution of multimodal features after concatenating the outputs of the FC2 and FC3 layers described in Figure 4.3. We plot the features on the first two principal components using principal component analysis (PCA). The images come from a separate camera used to record the trials, not the camera used by the execution monitor.

4.4 Evaluation

We evaluated our multimodal execution monitor with a robot-assisted feeding system that performs scooping and delivers a spoon of yogurt to the mouth of participants. We conducted our evaluations with approval from the Georgia Tech Institutional Review Board (IRB).

4.4.1 Instrumental Setup

Our robot-assisted feeding system uses a PR2 from Willow Garage, which is a general-purpose mobile manipulator. The PR2 consists of an omni-directional mobile base and two 7-DOF back-drivable arms with powered grippers. We run a $1\,\mathrm{kHz}$ low-level PID controller with low gains and a $50\,\mathrm{Hz}$ mid-level model predictive controller from [156] without haptic feedback. We designed 3D-printed handles so the PR2 can grip both a spoon and a bowl. We also affixed a spill guard and bars for wiping the spoon to the bowl. After scooping yogurt, the robot can drag the spoon across the bars to clean off yogurt from the bottom of the spoon (see Figure 4.6).

We mounted multiple sensors on the robot for multimodal sensing during feeding assistance. We mounted a force/torque sensor (ATI Nano25) between the handle and the spoon to measure the forces and torques applied to the spoon by the user at $1\,\mathrm{kHz}$. To estimate the location of a user's mouth, we mounted an RGB-D camera (Intel SR300) on the right arm's wrist (see Figure 3.10). We also use the Intel SR300's 2-channel microphone array to measure and localize sounds. To sense collisions with the robot's body, we covered the robot's left arm with fabric-based whole-arm tactile sensors introduced in [176]. These tactile sensors provide both contact locations and forces. Our monitoring system only uses the sum of all the estimated force magnitudes from the tactile sensors.

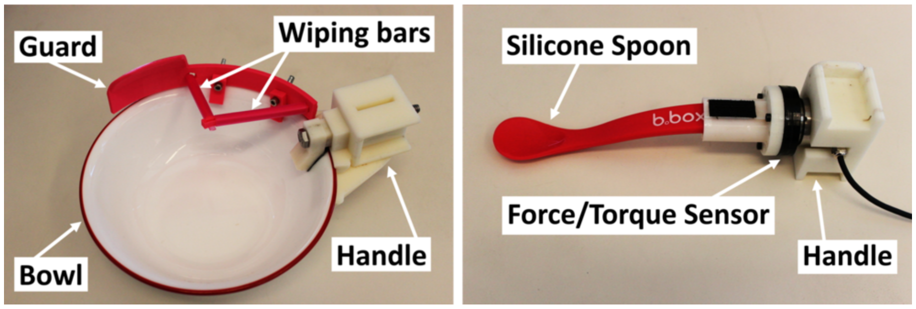


Figure 4.6: **Left:** A bowl with an attached handle, guard, and wiping bars to avoid spilling food. **Right:** A tool for feeding that has a flexible silicone spoon and force-torque sensor.

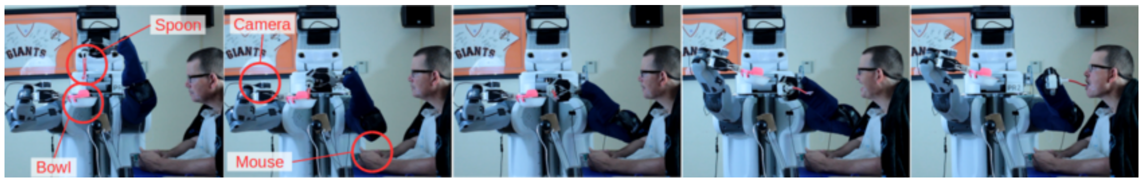


Figure 4.7: An image sequence of the entire scooping and feeding process with Henry Evans, a person with quadriplegia, in the living room of his home.

4.4.2 Robot-Assisted Feeding System

Our robot-assisted feeding system performs three autonomous subtasks (see Figure 4.7). A user is able to command the robot to perform the ‘scooping’, ‘clean spoon,’ and ‘feeding’ subtasks using a web-based graphical user interface (GUI). Given the ‘scooping’ command, a PR2 estimates the location of the bowl held by its right arm and then scoops a spoon of yogurt using predefined motions. To avoid spilling yogurt, the user can then command the ‘clean spoon’ subtask, which involves the PR2 dragging the back of the spoon over the wiping bars. Given the ‘feeding’ command, the robot estimates the user’s mouth location using the SR300 camera and then moves the spoon to the user’s mouth, inserts it, and retracts it. At any time during the feeding task, the user can stop the robot by clicking anywhere on the screen and then resume feeding by re-executing the previous subtask.

4.4.3 Simulated Anomalies

We asked able-bodied participants to produce 12 representative anomalies for the evaluations of our feeding and execution monitoring system. Figure 4.5 shows the anomalies produced by the participants. Prior to participants producing simulated anomalies, we showed a demonstration video of several possible anomalies and instructed them on what to do. We then encouraged participants to produce any of the anomalies at any time with any variation. When we collected data with Henry Evans, a person with severe quadriplegia, we asked his wife and primary caregiver, Jane Evans, to produce some of the anomalies after seeing the demonstration video.

4.4.4 Evaluation Process

We first evaluated our monitoring system with 8 able-bodied participants. We recruited able-bodied participants—3 males and 5 females—whose ages ranged from 19 to 35. They were all novice users who did not have any experience with our feeding system. Each participant performed 20 non-anomalous and 24 anomalous executions over a total of 1.5 hours in a closed experiment room. The 24 anomalous executions consisted of all 12 anomalous cases being recorded two times. In total, we collected data from 160 non-anomalous and 192 anomalous feeding trials. During non-anomalous executions, we asked participants not to move their upper bodies and arms to approximate a lack of movement due to disabilities. To quantify the performance of our system, we performed leave-one-person-out cross-validation by training our execution monitor with 7 participants' data and testing the monitor with the 1 participant remaining.

We also included 508 hand-labeled images, which includes 12 anomalies, from 2 extra participants—average 33 years old of 2 females—who were not included in the above dataset but were recorded when piloting the experiment. We also performed data augmentation in which we added Gaussian random noise to individual signals for the temporal features and random rotations, translations, and magnifications of images for the convolutional features. We trained and tested our execution monitor using an Amazon EC2 server with 4 cores, 61GB of memory, and an NVIDIA K80 GPU.

We also tested our system with Henry Evans at his home in California, USA over a span of 4 days. This was our first test of our system with a person with disabilities. Our lab has an ongoing long-term collaboration with Henry Evans and his wife and primary caregiver, Jane Evans. Henry Evans is severely impaired, but he can move his head and a finger sufficiently well to operate our system's web-based GUI using an off-the-shelf head tracker and a mouse button. Henry is unable to speak and has difficulty eating. He frequently eats yogurt, which he specifically requested when we initiated our work on robot-assisted feeding. On the first day, we began with safety training and then allowed Henry to practice with the feeding system until he became comfortable using it. Henry then performed 20 non-anomalous feeding trials during which he was able to successfully eat yogurt each time. For the following three days, he participated in 5 sessions during which we used the anomaly detection and classification systems trained on the data from the 8 able-bodied participants. In each session, Henry performed 10 non-anomalous and 12 anomalous yogurt feeding trials in random order for about 1 hour in total. A caregiver, Jane Evans, produced 'sound by user' and 'touch by user' for him.

4.4.5 Baseline Methods for Anomaly Classification

Our proposed method consists of an MLP with both temporal and convolutional features, or MLP(T+C). Below we present six alternative methods for performing anomaly classification. We compare each of these methods to our approach in Section 4.5. Note that we ran each classification method with the same anomaly detector.

- Random: This method randomly determines the type and cause of anomalies.
- SVM(R): A support vector machine (SVM) classifier with a radial basis kernel. Type and cause of anomalies are determined with raw data used for temporal features.
- SVM(H): The same SVM structure with histogram of oriented gradients (HOG) features extracted from images.
- SVM(T): The same SVM structure with temporal features.

- MLP(T): MLP with only temporal features.
- MLP(C): MLP with only convolutional features.

4.5 Results and Discussion

We first investigated how the use of multimodal signals helps to classify anomalies. Figure 4.8 shows the distribution of multimodal signals observed from the robot-assisted feeding tasks with 8 able-bodied participants. The blue region shows the mean and standard deviations of 7 features from 160 non-anomalous feeding executions. We can observe clear patterns that were subsequently used to train the HMMs for anomaly detection. The red curves show an anomalous execution in which a spoon collided with a user's mouth due to a system fault. We can observe a short bump in 'force on spoon' from the collision around 1.5s. The unexpected change may be sufficient to detect the anomaly, but it is difficult to estimate its cause among the other causes. Instead, as we can see in Figure 4.5, the use of multimodal sensory signals helps separate anomalous events into different regions.

We also evaluated the overall effectiveness, or accuracy, of our execution monitoring system. To do so, we used the feeding data from the 8 able-bodied participants and performed leave-one-person-out cross validation. Our anomaly detector achieved 83.27% accuracy in detecting anomalous trials throughout all 352 feeding trials. We then used all anomalous data from the 8 able-bodied participants to train and test our anomaly classifier. Figure 4.9 shows the comparison of our proposed method against the 6 baseline methods. Our proposed method had a classification accuracy of 81.37% which was 5% higher ($p = 0.0097$ from t test) than the next best method, SVM(T). Compared to the MLP(T), our proposed method resulted in 17% higher accuracy due to the inclusion of the convolutional features.

Figure 4.10 shows a confusion matrix for our classifier's performance with respect to the 12 known anomalies. Our proposed method successfully determined the causes of most anomalies except for the 'system freezing' event. In this work, we reproduced 'system

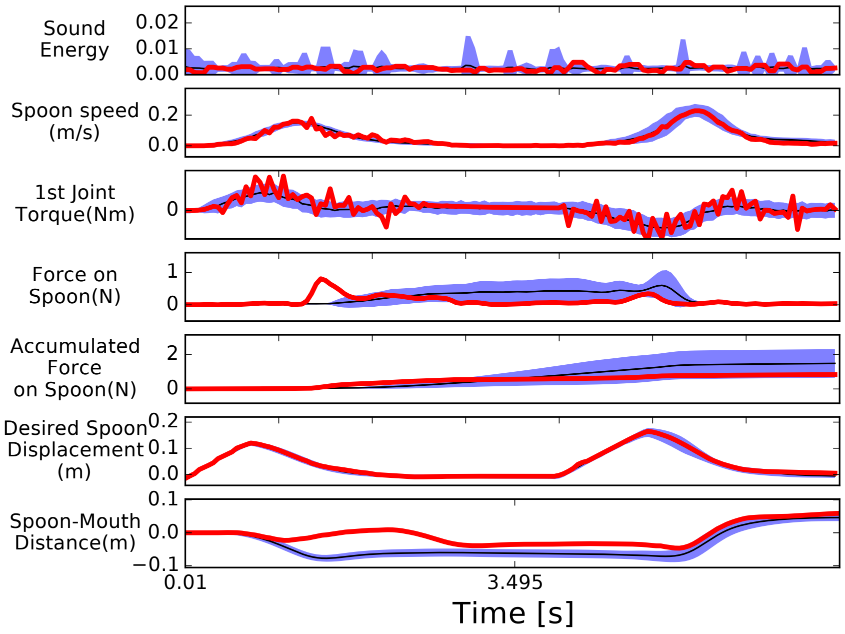


Figure 4.8: Multimodal sensory signals used in the anomaly detection. Blue regions show their mean and standard deviation from non-anomalous feeding executions. Red curves show the signals from an anomalous feeding event: spoon collision due to a system fault at $1.5\,\mathrm{s}$.

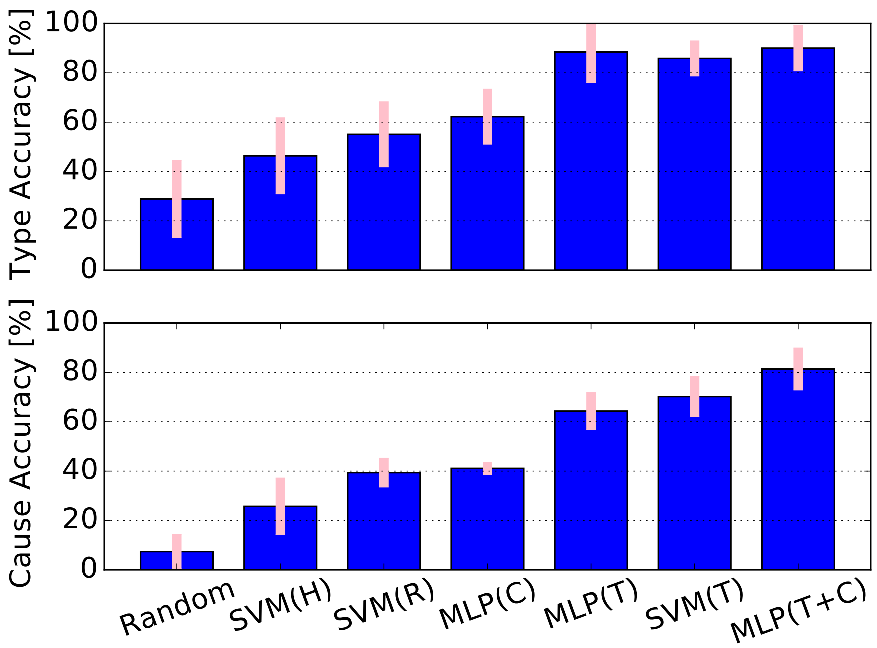


Figure 4.9: Comparison of classification methods on 8 able-bodied participants' feeding data. The two bar charts show the detection accuracies with respect to the type and cause of anomalies described in Figure 4.2. The symbols in parentheses indicate the type of input features (R=Raw signal data, C=convolutional features, H=HOG features, T=temporal features).

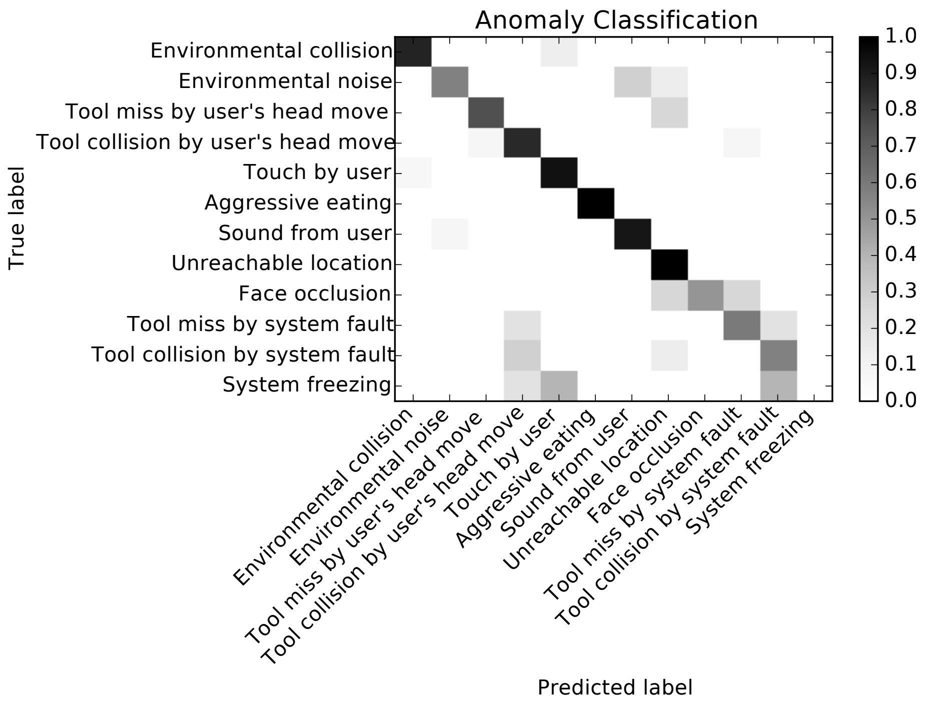


Figure 4.10: Confusion matrix from the proposed anomaly classifier applied to data from 8 able-bodied participant using leave-one-person-out cross validation.

freezing' by randomly killing the model predictive controller process while the robot was moving its arm to a location. However, our classification success for this event was limited by the fact that our system only successfully detected the anomaly 5 out of 16 times in training. This may be due to the lack of effective features since no signal changes while in the freeze status, except 'desired spoon displacement' and our method did not monitor sensory streams related to the internal state of the robot.

In our first test with Henry Evans, he successfully fed himself with the robot for all 20 consecutive trials. That is, he ate 20 scoops of yogurt produced by Chobani, LLC. We then evaluated the execution monitor through 5 additional sessions, each of which included 10 non-anomalous and 12 different causes of anomalous executions in random order, for a total of 50 non-anomalous and 60 anomalous executions. Our monitoring system achieved 86.36% detection accuracy over all 110 executions with Henry. Our system also successfully detected two unintentional real anomalies caused by researcher mistakes and camera faults in the new environment. Our classifier successfully determined the types of anomalies with 90.51% accuracy (i.e., 'tool collision,' 'tool miss,' 'sound,' or 'body collision'). However, it classified the specific causes of anomalies with only 53.44% accuracy, which is roughly 30% lower than the cross-validation accuracy in our lab environment. In this evaluation, we did not train on any data collected from Henry.

Notably, our results with Henry Evans only used training data from able-bodied users in a controlled laboratory setting. The system's overall performance generalized reasonably well given that we conducted this test in Henry's home and that his impairments influence the way he eats. We expect that training on user-specific data and on more data with greater variation could improve results for in-home use.

CHAPTER 5
APPLICATION: A ROBOT-ASSISTED FEEDING SYSTEM

5.1 Overview

Activities of daily living (ADLs), such as feeding, toileting, and dressing, are important for quality of life [1]. Yet for many people with disabilities, including the people with upper limb disabilities or quadriplegia, such tasks prove challenging without assistance from a human caregiver. However, the shortage of healthcare workers and rising healthcare costs create a pressing need for innovations that make assistance more affordable and effective.

Numerous specialized assistive devices, including specially designed robots, have been developed to help people with disabilities perform ADLs on their own [177]. Each device cost varies and typically provides a narrow form of assistance suitable for people with particular impairments. Alternatively, researchers have developed general-purpose mobile manipulators, proof-of-concept platforms that focus on various applications: rescue, assistance, residential service, etc. The robots often have a mobile base and two (or one) human-like arms (ex., PR2 robot from Willow Garage and Jaco arm from Kinova), and enable users to overcome their physical or perceptual limitations. Although the robots have the potential to provide a wide variety of assistive tasks [119], their complexity creates challenges, including the risk of being too difficult to use.

A representative example is feeding that is an essential ADL and important for staying healthy. People with upper-limb disabilities, including individuals with quadriplegia, are incapable of feeding themselves. Currently, a number of commercial feeding devices (or robots) are available: My Spoon [127], Bestic arm [128], Mealtime partner [129], etc. These specially designed devices often have a (desk-mountable) fixed base, low degree-of-freedom (DoF) arm, and limited sensing capabilities. Most robots provide *passive feeding*, in which a robot delivers food into a pre-defined location in front of the mouth and requires a user's upper limb movement to take the food. The robots also often require users' physical interactions such as placing and teaching. Alternatively, we can use a general-purpose mobile manipulator as a robotic-feeding system with its physical and sensing abilities for diverse users.

In this dissertation, as a test bed of the monitoring system, we introduce a robot-assisted feeding system that enables a general-purpose mobile manipulator, a PR2 robot, to assist people with disabilities with safe, easy-to-use feeding executions. The system can perform three independent tasks: scooping (or stabbing), wiping, and feeding. A user can command a preferred task via a graphical user interface (GUI). The system performs *active feeding* in which the PR2 delivers food inside a user's mouth after scooping or stabbing food using a user-selected utensil. Our previous work that is a part of this dissertation introduced a proto-type of an *active feeding* feeding system with ARTags [8]. We improved the system by autonomously detecting the food and the user's mouth without the tags.

Our feeding system contributes to the adoption of a general-purpose mobile manipulator as an assistive robot. In details, we design the system so that it satisfies a list of designing factors for feeding devices in literature: convenience, comfort, speed, and safety as well as food grasping and delivery functions [178, 133]. We enable the system to delivery various soft or solid foods using 5 different utensils and 2 types of bowls. We also provide software and hardware interfaces for users to easily register or exchange a utensil and a bowl. The system also allows the users to access its interface from any web browsing devices. For comfort and safety, the system provides safe and comfortable food delivery by using a low-gain controller and a multimodal execution monitor to check failures.

5.2 Outline of System

5.2.1 System Configuration

Figure 5.1 shows the configuration of our robot-assistive feeding system. We use a general-purpose mobile manipulator, a PR2 robot from Willow Garage. The PR2 is a 32-DoF mobile manipulator that consists of an omni-directional mobile base, a 1-DoF telescoping spine, and two 7-DOF back-driverable arms that are controlled by $1\,\mathrm{kHz}$ low-gain PID controllers. Its maximum payload and grip force are listed as $1.8\,\mathrm{kg}$ and $80\,\mathrm{N}$, respectively. We run the PR2 on Robot Operating System (ROS) Indigo.

The system can perform three independent tasks: scooping (or stabbing), feeding, and wiping. A user can command a preferred task via a GUI (see Figure 5.2). For the scooping, the system finds the best scoopable (or stabbable) location using a head-mounted RGB-D camera, Microsoft Kinect V2, to scoop a spoonful of food in a bowl held by its right arm. For the *active feeding*, the system estimates the user's face and mouth pose by mounting an Intel SR300 RGB-D camera on the top of the right wrist.

In addition to the cameras, the system runs various sensors to check anomalous behaviors during the tasks. The wrist-mounted camera has a 2-channel microphone that can simultaneously record audio at a $44.1\,\mathrm{kHz}$ sampling rate. The system collects contact force using a force-torque sensor attached on the utensil holder and fabric-based tactile skin on both arms. We will discuss how to use these multimodal sensory signals to determine anomalous behaviors in Section 5.3.4.

5.2.2 Operating Procedure

We will explain the operating procedure when a person with disabilities wants to eat yogurt. We assume the robot is already placed at a location reachable from the user's mouth and holding a utensil and a bowl. We also assume the user can move and click a mouse pointer using a finger or a head (or eye) tracker. A list of typical operations are as follows:

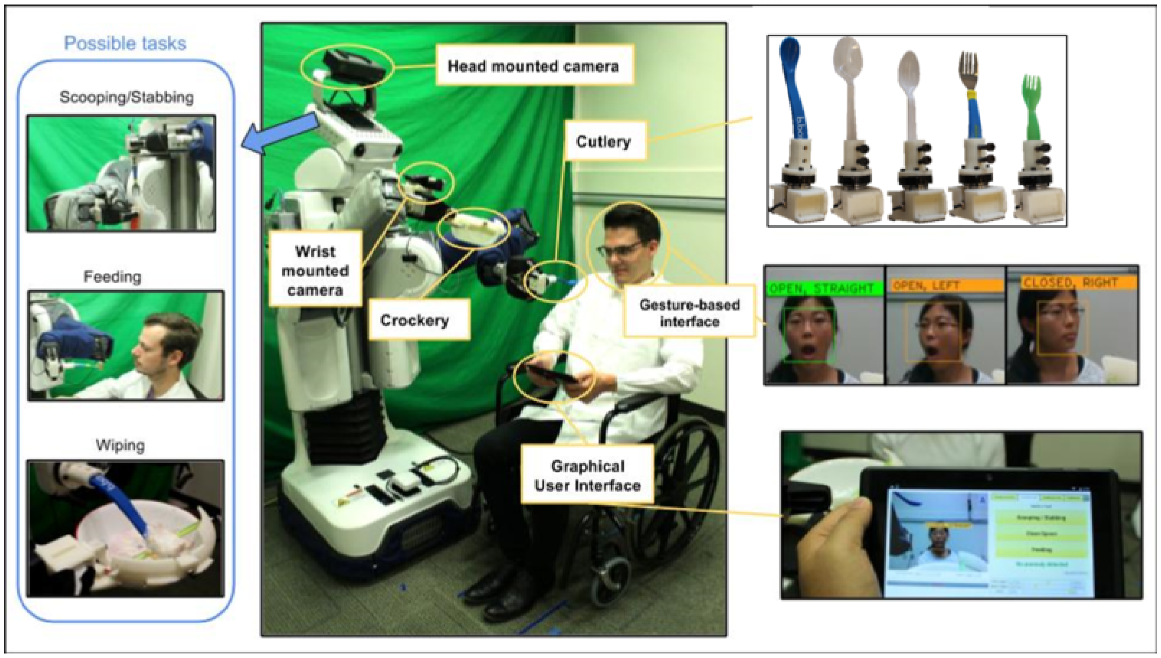


Figure 5.1: The outline of a robot-assisted feeding system.

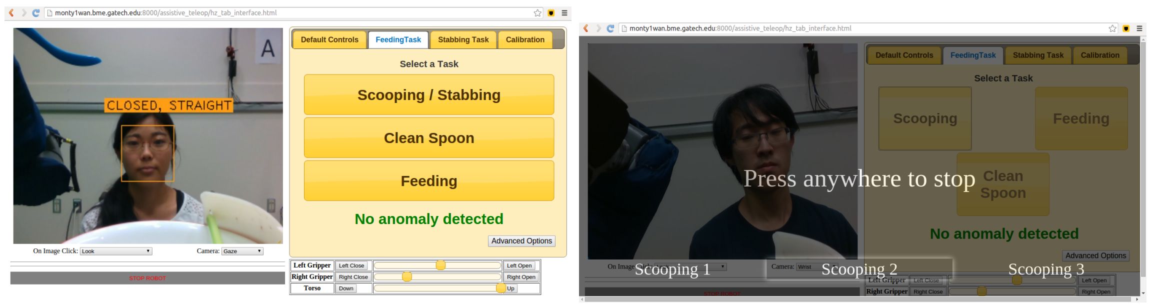


Figure 5.2: **Left** A web-based graphical user interface with a feeding tab. **Right** A full-screen stop button.

- **Scooping**: The user clicks the Scooping button on the GUI. The robot then scoops a spoonful of yogurt from the bowl using pre-defined motions.
- **Wiping**: The user clicks the Wiping button if the scooping task brings excessive food on the top of the spoon or a residue at the bottom of the spoon. The robot wipes off the surface of the spoon using the wiping bar on the bowl (see Figure 5.5).
- **Feeding**: The user clicks the Feeding button when an adequate amount of yogurt is present on the spoon. The user turns his or her head toward the camera on the robot's right wrist. The robot then estimates the pose of the user's mouth and deliveries

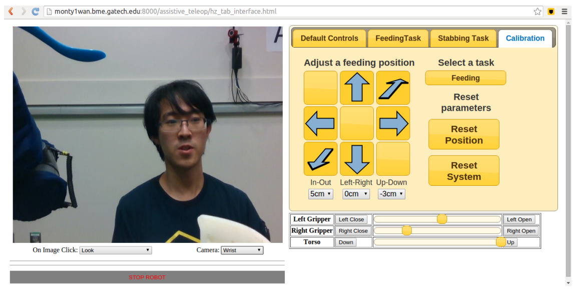


Figure 5.3: A feeding-position calibration tab on the GUI. Users can add an offset to their desired direction.

yogurt inside of the mouth.

During the tasks, the user can stop and run the robot again whenever they want to use the provided interfaces.

Our multimodal execution monitor runs in parallel with the scooping and feeding tasks. When it detects an anomalous execution that largely differs from typical non-anomalous executions, the PR2 immediately forces the system to stop and move its arm back to the initial pose.

5.3 System Description

5.3.1 User Interface

Our system uses an web-based GUI platform developed for self-care tasks around the users' head [2]. We modify the platform to transmit task commands, display visual outputs from the cameras, and collect feedback from users (see Figure 5.2). The GUI uses a *rosbridge* ROS library [179] that allows web browsers to interact with the ROS using ROS topics and services over web-sockets. We could develop the necessary interfaces on the top of the platform using HTML5, CSS3, and JavaScript. Any users can access the interface using web-browsing devices such as a tablet or a laptop with their own pointing devices such as a mouse and a head tracker. The interface consists of a live video screen that displays the video output from the head- or wrist-mounted cameras and a task tab that provides buttons or bars to command a task or adjust internal parameters of the system.

In the feeding task tab, the users can select one of three task buttons: Scooping/Stabbing, Clean spoon, and Feeding. The system then executes the selected task till finishing the task or receiving a stop command. The users can force the robot stop at any moment by clicking a full-screen stop button that appears during any task executions. The stop command is treated as an anomalous event which triggers a corrective action following transition T_A in finite-state machine (FSM) described in Section 5.3.2. After the task completion, the users may then enter feedback (i.e., *success* or *failure*). We used the feedback to label the current execution data to train/test the execution monitor and tune the performance of the system.

In addition, users can select a comfortable feeding location where the robot places the entire spoon or fork with food inside their mouth. In default, the robot place the tip of utensils $4\,\mathrm{cm}$ inside from the center of the mouth plane (Red-Green plane in Figure 5.9). Figure 5.3 shows a feeding-position calibration tab with 6 arrow buttons to add $\pm 1\,\mathrm{cm}$ offset into the selected direction.

5.3.2 Task Executions

Finite-state Machine Figure 5.4 shows the flow of the overall task sequences using a finite-state machine (FSM). To perform the scooping task, the robot first initializes its configuration to a scooping pose and estimates the location of the bowl. After the selection of food location, it then approaches to scoop a spoonful of soft food or stab a chunk of solid food. The user can then select to proceed to the feeding task if an adequate amount of food is present on the spoon. Otherwise, the user can command the robot to re-scoop food or wipe off the spoon to remove excessive food.

We incorporated two transition triggers to switch between states. Shared autonomy is provided in which the user's input can trigger the T_A and T_N transitions via the web-based GUI or the vision-based gesture detector. The multimodal execution monitor in Section 5.3.4 can also trigger an *anomalous* transition. When an anomaly is detected, the system transitions to a state in which the robot halts or performs a corrective action. For instance,

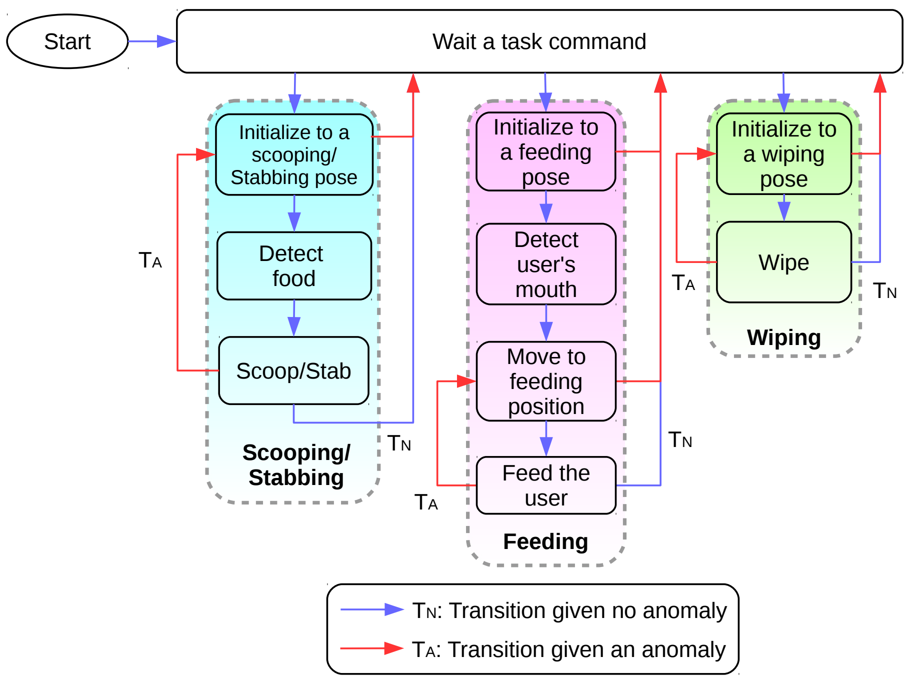

Start
Wait a task command
Initialize to a scooping/ Stabbing pose
Detect food
Scoop/Stab
Scooping/ Stabbing
TA
TN
Initialize to a feeding pose
Detect user's mouth
Move to feeding position
Feed the user
Feeding
TA
TN
Initialize to a wiping pose
Wipe
Wiping
TA
TN
TN: Transition given no anomaly
TA: Transition given an anomaly

Figure 5.4: Finite-state machine for the robot-assisted feeding. Each box represents a state, whereas arrows indicate state transitions.

if a loud and unexpected sound is detected while feeding, an anomaly is triggered and the robot will retract its arm to avoid harming the user.

Scooping / Stabbing Task The scooping and stabbing tasks aim to pick and hold designated food. Users can mount a pair of utensil and bowl for various type of foods. Figure 5.5 **Right** shows 5 representative utensils we used in our evaluation: a silicone spoon, small/large plastic spoons, a plastic fork, and a metal fork. A user (or a caregiver) is able to mount a preferred utensil on the 3D-printed tool-changer after registering the transformation information from the changer to the utensil tip. Currently, the system provides a YAML file for an exper user or researcher to register a new utensil. There is also a 3D-printed handle the robot can firmly grasp. Figure 5.5 **Left** shows a bowl our robot typically holds in evaluation. We also attach another handle on the bowl to enable a PR2's grasp.

Food spilling frequently occurs during the scooping or stabbing tasks due to an excessive amount of food and imperfect manipulation. To prevent the spilling from the bowl, we

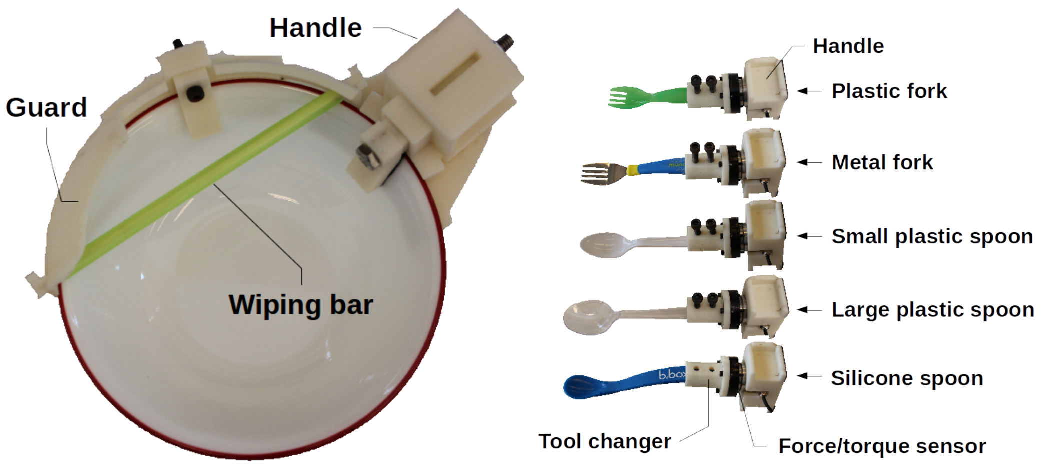


Figure 5.5: **Left:** A bowl with an attached handle and guard/wiping bar to avoid spilling food. **Right:** Five utensils consisting of a flexible silicone spoon, small/large plastic spoons, a plastic fork, and a metal fork attached on the top of a force-torque sensor.

mount a 3D-printed bowl guard (see Figure 5.5 **Left**). We also place a cylindrical bar to wipe off excessive food from the bottom of the spoon.

The system produces either scooping or stabbing motions using a sequence of motion primitives. It is also able to adapt (i.e., translate) its motion with respect to food location and the transformation information from the changer. In this work, the robot can vary the scooping location by visually estimating the best scoopable or stabbable point or randomly selecting it inside the bowl. For the visual selection, we use a food location estimator described in Section 5.3.3. Otherwise, the robot approaches a random location inside the bowl. Note that we pre-registered the size of bowl so the system is able to estimate the range of scooping/stabbing area based on current kinematics information.

Feeding Task The feeding task aims to provide safe and easily accessible feeding service to a wide range of users such as people with severe quadriplegia. Unlike conventional *passive feeding* systems in literature, our *active feeding* system does not require user's upper body movements to take food on the utensil. In other words, our system automatically delivers food inside the user's mouth.

To put food inside the mouth, we use a 3D mouth-pose estimator using the wrist-mounted RGB-D camera that allows the robot to observe the user's frontal face. We will explain this detail in Section 5.3.3. After estimating the mouth pose, the system determines a feeding position inside the mouth with a pre-defined or user-selected offset (i.e., $4\,\mathrm{cm}$ in default). After the first delivery, the robot stores and re-uses the estimated mouth pose to shorten the delivery time. Our system also provides a button to re-estimate the mouth pose when the user wants (see Figure 5.3). We will explain available options for users in Section 5.3.1.

Our system does not require pose teaching by users or caregivers. The system has a set of pre-defined trajectories, linearly interpolating pre-defined poses in a pre-defined mouth coordinate frame that is visualized in Figure 5.8. After estimating the new mouth pose, the system transfers the trajectories from the new mouth coordinate to the world coordinate. Note that our system provides spoon tilting motions for the large plastic spoon since users have a difficulty to scrape the stiff and deep spoon against the upper lip to take food off from the spoon.

For safety, the system observes the user's face and the spoon during the feeding. However, the robot may not fully observe the face due to the occlusion by another end-effector and its spoon. Our system lifts up the camera to avoid occlusion and then the landmark estimator is able to predict points on the small occluded area (see Figure 5.6). In addition, the system is able to tilt the spoon tip during the feeding. This helps the user to eat food on the deep spoon comfortably.

After each scooping, the user can confirm whether yogurt is present on the spoon. If there is not an adequate amount of yogurt on the spoon, the user can instruct the robot to re-scoop some yogurt. Finally, the user can provide feedback to the system after each task, which labels collected data to train the anomaly detector described in Section 5.3.4.

Wiping Task The scooping task often brings an excessive amount of food on the top of the spoon or a residue left at the bottom of the spoon. Both may result in food spills during

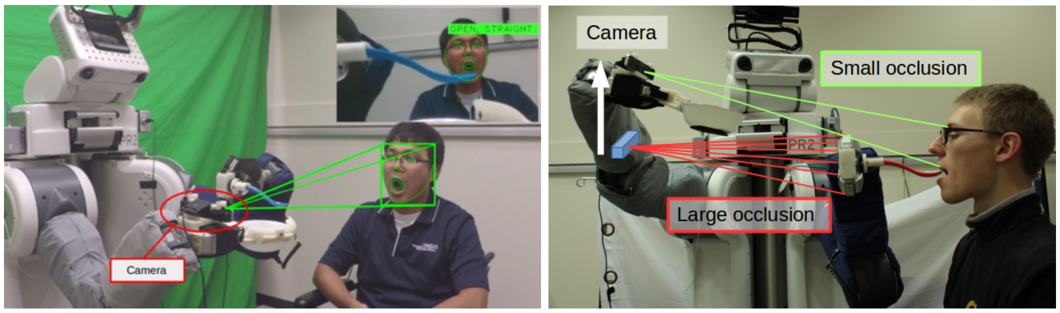


Figure 5.6: Visualizations of facial landmarks and occlusions. Our system estimates the landmarks from a part of face occluded by a spoon tip or hand. To minimize the occluded area, the system also lifts up the wrist-mounted camera during the feeding.

the delivery and discomfort during the feeding. A number of commercial robots use the edge of a bowl to clean excess food. Meal Buddy, particularly, uses a wiping bar to wipe off excess food from the spoon and wipe drips from the bottom [130]. Our system also solves these issues by attaching 3D-printed food guard and wiping bar on the bowl (see Figure 5.5 **Left**). The guard is $3\,\mathrm{cm}$ high, and used to block food spills from the bowl while performing the scooping motions. Likewise, the bar is $13.5\,\mathrm{cm}$ long, and used to remove a residue at the bottom of the spoon when the user commands the wiping task by clicking a Clean Spoon button on our GUI. The robot drags the bottom surface of the spoon on the bar following a pre-defined linear trajectory. Note that the relative displacement between the right end effector and the bar is fixed due to the rigid grasping of the bowl.

5.3.3 Estimation Modules

A Food-location Estimator Our food-location estimator finds a location where the robot can scoop (or stab) the largest amount of food from the bowl. After repeated scooping (or stabbing) executions, the robot may fail to take remnant food. It requires caregivers' intervention for refilling or relocating food. A commercial feeding robot, iEAT [180], rotates its food plate to adjust the scooping point. Alternatively, we solve this issue by developing a vision-based estimator. We set 5 scoopable locations displayed as yellow markers in Figure 5.7, in the bowl. Given the estimated center and pre-defined diameter of

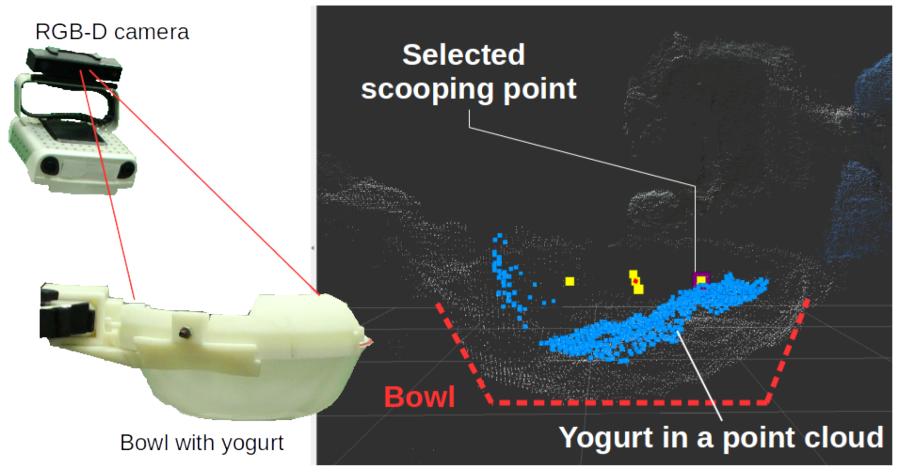


Figure 5.7: A food-location estimator finds the best scoopable location from a point cloud. Blue points indicate the surface of foods inside a bowl. Yellow and red points show 5 potential scooping (or stabbing) locations and a selected best location, respectively.

the bowl, the estimator first extracts a point cloud that corresponds to food inside the bowl using the head-mounted RGB-D camera. It then clusters the point cloud using a Gaussian kernel with the 5 locations as cluster centers. It finally selects a location with the highest density and sends it as the scooping (or stabbing) location to the robot.

A Mouth-pose Estimator The estimation of a user's mouth pose is one of the main component to autonomously and robustly provide this feeding assistance to diverse users. In our previous work [8], the robot first determined a location for the user's head by detecting an ARTag attached to the user's forehead and transformed the location into the user's mouth using a predefined offset (as seen in Figure 5.8). In this dissertation, we extend the work by introducing a mouth-pose estimator that does not require the ARTag and the offset by directly estimating the location of the user's lips from an RGB-D image.

Our estimator first extracts facial landmarks, key points of interest to localize facial regions such as mouth, nose, left eye, right eye, and jaw, using the open source dlib library [159, 160]. This process localizes a user's face from an RGB image and detects 68 landmarks for a frontal face. We then convert these to 3D points by projecting on a depth image. This process may produce large errors due to the noise in the depth data or approximated

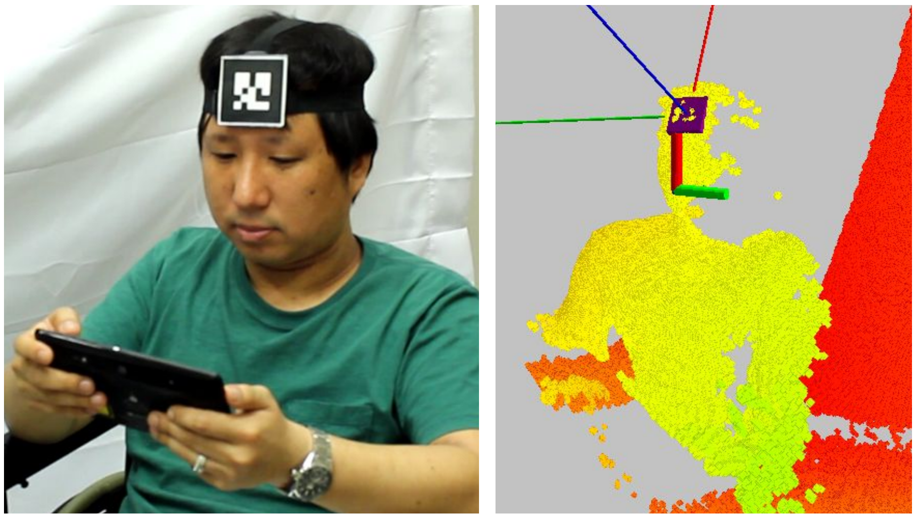


Figure 5.8: Our previous system [8] estimated the location of the user's mouth using a Kinect V2 and an ARTag attached to the user's forehead.

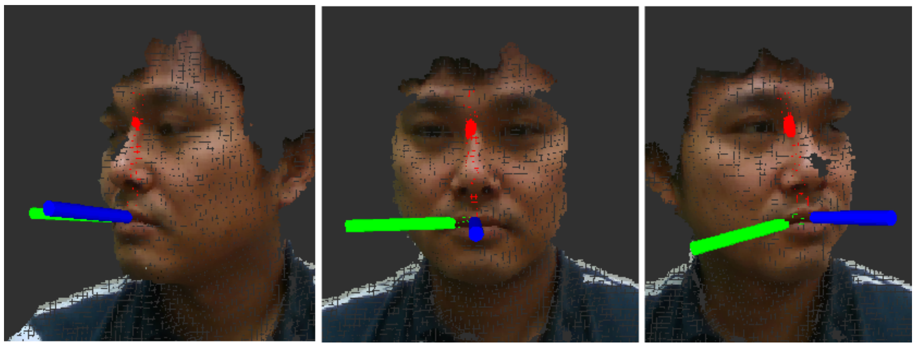


Figure 5.9: Estimated mouth poses from our vision-based mouth pose estimator.

time-synchronization of RGB and depth images. Thus, we reject landmarks 1) giving large interdistance between close landmarks and 2) giving large displacement ($\leq 5\,\mathrm{cm}$) from last landmarks. We also reject a set of landmarks around eyes that largely differ from pre-registered landmarks around eyes based on [181]. After the rejection process, we apply the Delaunay triangulation of the landmarks and group these into three groups: cheek, eye, and mouth. The estimator then computes the positions of the cheek, the eyes, and the mouth using their geometric relation similar to Schrer et al.'s work [5]. These three positions are used for computing the orientation of the mouth.

5.3.4 Safety Tools

The system may produce anomalous behaviors due to task, user, environmental failures. This section introduces various levels of safety tools for the robot-assisted feeding system to prevent or reduce hazards.

Hardware The PR2 arms are backdriveable and controlled by a low-gain PID controller so the system can reduce the magnitude of unintended collision. The PR2 also provides an emergency stop button that a user or a caregiver can hold and cut off power to the PR2's motors with. While cutting the power off, the passive spring counter balance system makes the arms float so it is able to prevent unintended collision.

Software Our GUI provides a full-screen stop button (see Figure 5.2 **Right**) for people with disabilities to conveniently and quickly cancel the current task and force the robot stop. During task executions, the button expands to the entire web browser and, if the user clicks anywhere on the browser, the *rosbridge* server of GUI sends a stop command to the system. The command then triggers the T_A transition on FSM. In the end, the robot returns to the initial pose of the current task.

Another safety tool is the multimodal execution monitoring system described in previous chapters. The monitoring system detects anomalous robot executions using multiple sensors. Figure 5.10 shows the sensors mounted on the PR2: SR300 RGB-D camera with two-channel microphone, joint encoder & current sensor, fabric-based tactile skin sensor, and ATI force/torque sensor. Continuously collecting sensory signals from the sensors, our monitor checks the scooping (or stabbing) and feeding tasks in online. Figure 5.11 shows the framework of the multimodal anomaly detector. The detector determines an anomaly when the pattern of incoming sensory signals largely differ from the typical pattern from past successful feeding executions [42]. It immediately forces the system to stop and move its arm back to the initial pose for safety following the FSM. The classifier then determines

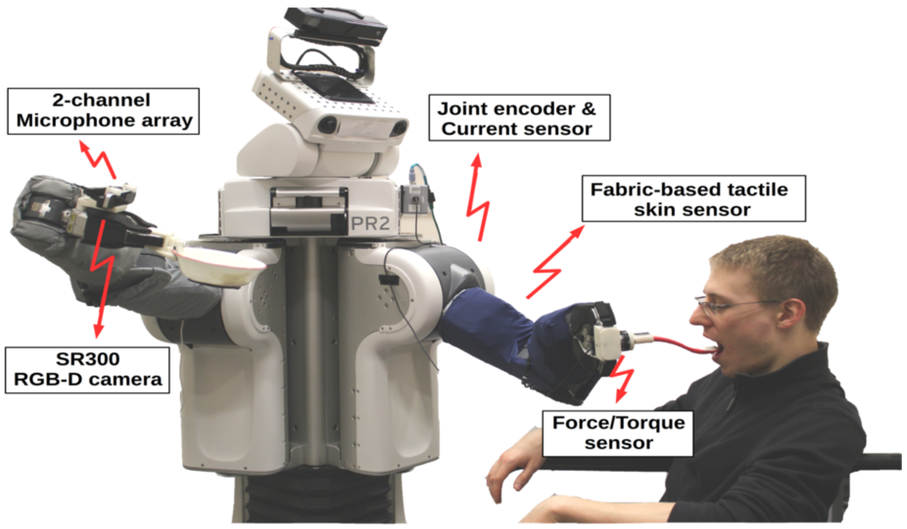

2-channel Microphone array
Joint encoder & Current sensor
Fabric-based tactile skin sensor
SR300 RGB-D camera
Force/Torque sensor
PR2

Figure 5.10: A variety of sensors we used to monitor task executions. This figure shows a feeding demonstration with an able-bodied person.

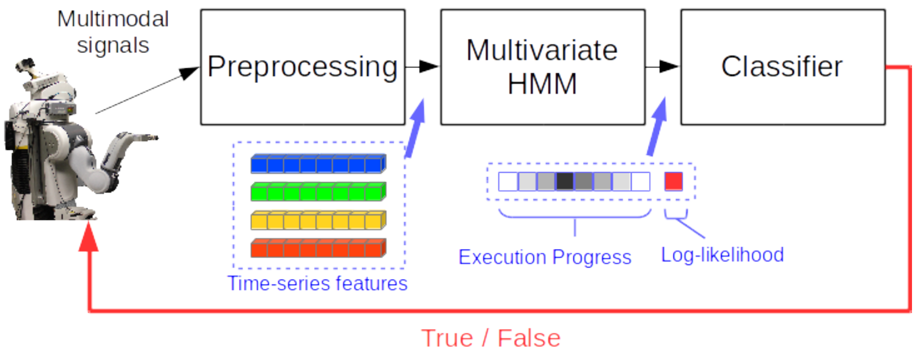


Figure 5.11: Anomaly Detection Framework using hidden Markov models.

the type and cause of the anomaly searching similar cases from an anomaly database [163]. The resulted information is sent to the robot system for correcting current executions, planning a recovery strategy, or improving the robotic assistance system.

5.4 Experimental Setup and Results

We evaluated the robot-assisted feeding system with able-bodied participants and people with disabilities. We conducted the evaluations with approval from the Georgia Tech Institutional Review Board (IRB). In this section, we address the experimental setup and results.

5.4.1 Evaluation with Able-bodied Participants

9 Able-bodied Participants As a step towards use by people with disabilities, we recruited 9 able-bodied participants and performed evaluations in our laboratory from April 30th to May 12th, 2017. They are all Georgia Tech students consisting of 1 female and 8 males with an average age of 25.4. We equally divided the number of participants into 3 groups where each group of participants feed themselves with *cottage cheese and silicone spoon*, *watermelon chunks and metal fork*, and *fruit mix and plastic spoon*, respectively. With the PR2 holding a bowl of the selected food and a utensil, participants controlled the robot to scoop (or stab) and feed themselves through an $7\,\mathrm{inch}$ Google Tango tablet with a Chrome browser. Before starting this evaluation, we briefly trained the participants to use the feeding system. As part of this training, they practiced using the system three times. Each practice run took about one minute. For safety, we held the emergency stop button during this evaluation.

Each participant performed 60 non-anomalous and 36 anomalous feeding executions $(= (20 + 12) \times 3$ sessions) for 3 hours. In this section, we use the non-anomalous feeding executions to evaluate the system. The participants performed total 540 non-anomalous executions and 19 extra executions. Note that they freely controlled the robot to wipe off the bottom of the spoon before they eat food. We also disabled the monitoring system not to generate false alarms during this evaluation. We recorded available sensory signals using a *rosbag* package and also recorded experiment videos. The participants also answered 11 post-experiment questions (five-point Likert type questionnaire items) after the experiment.

Table 5.1 shows the 11 questions answered by all 9 able-bodied participants. The right two columns represent the average and standard deviation of the five-point Likert-item scores where 1 and 5 indicate strongly disagree and strongly agree, respectively. The participants were mostly familiar with robotic applications and successfully able to eat food using the system with 4.89 and 4.67 average scores, respectively. The system was reported as safe and easy-to-use with 4.22 and 4.0 scores, respectively. However, the participants

Table 5.1: Five-point Likert type questionnaire items for 9 able-bodied participants. The last column provides the average and standard deviation of scores with 1=strongly disagree, 2=disagree, 3=neither, 4=agree, and 5=strongly agree.

Number	Questions	Score	
		Avg.	Std.
1	I am familiar with engineering.	4.78	0.67
2	I am familiar with robotic applications.	4.89	0.33
3	I successfully ate food using the system.	4.67	1.0
4	I am satisfied with using the system.	3.89	0.93
5	The system was easy-to-use.	4.0	1.0
6	I felt safe while using the system.	4.22	0.83
7	I was comfortable while using the system.	4.33	0.71
8	The system delivered an adequate amount of food.	3.0	1.58
9	The system delivered food with adequate speed.	3.11	1.69
10	The system accurately placed food in my mouth	4.0	0.87
11	The system provides sufficient safety tools or functions to prevent hazards.	4.56	0.89

neither agree or disagree with the adequate amount of food the system delivers and its delivery speed.

The Author We also designed a long-term evaluation to observe the system's daily assistance capability. The author conducted a total of 428 feeding executions for 22 days between April 3th and July 28th, 2017 (see Figure 5.12). The author ate 6 types of foods (i.e., yogurt, rice, fruit mix, watermelon chunks, cereal, and cottage cheese) and 5 utensils (i.e.,small/large plastic spoons, a silicone spoon, and plastic/metal forks) in laboratory. Figure 5.13 shows the 6 examples of foods we used in this evaluation. We used the system until each day's food ran out so the number of executions varied. The author determined the success and failure of an individual task. Note that we turned on the execution monitor with the empirically determined low sensitivity only the yogurt feeding.

Figure 5.14 shows the number of daily successful and failed executions during the long-term self evaluation. The participant selected a food (i.e., yogurt$\times 6$, rice$\times 6$, fruit$\times 6$, cereal$\times 1$, cheese$\times 3$) and an adequate utensil among the 5 utensils. We recorded what

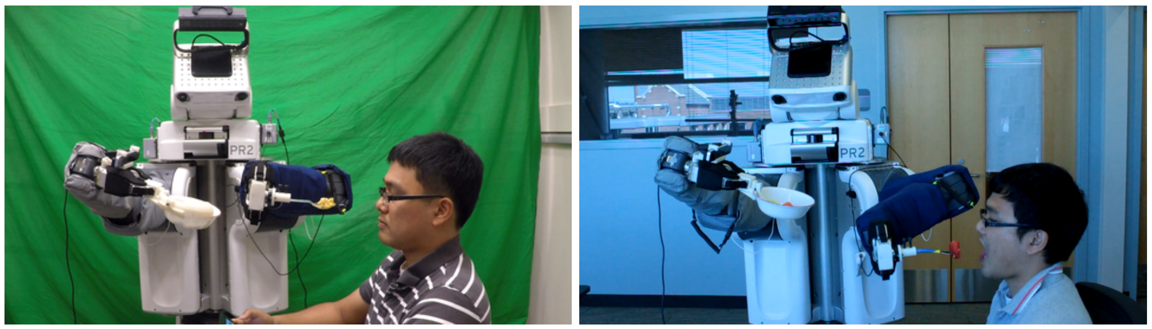


Figure 5.12: Captures of the long-term evaluation by the author. Our feeding system fed 6 types of foods to the author for 22 days.

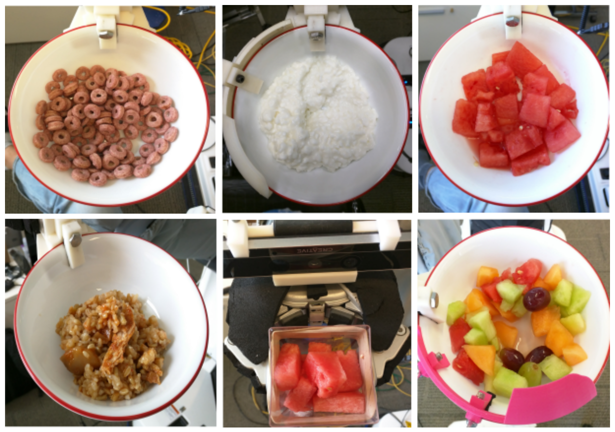


Figure 5.13: Examples of foods we used during the long-term evaluation.

kinds of failure occurs. Throughout this evaluation, we confirmed the system resulted in average 16 (3.6%) of feeding failures out of total 444 executions due to *a camera fault*, *false alarms from the execution monitor*, *tool collisions by system fault*, *a system freeze*, and *unknown reasons*.

5.4.2 Evaluation with People with Disabilities

We also conducted evaluations with Henry Evans and 3 people with disabilities. Henry became quadriplegic and mute after a stroke-like attack in 2003. As a main collaborator of our laboratory, he has participated in our assistive robot studies from 2010. We also recruited the 3 participants who have difficulties in eating using hand-held utensils from November, 2017.

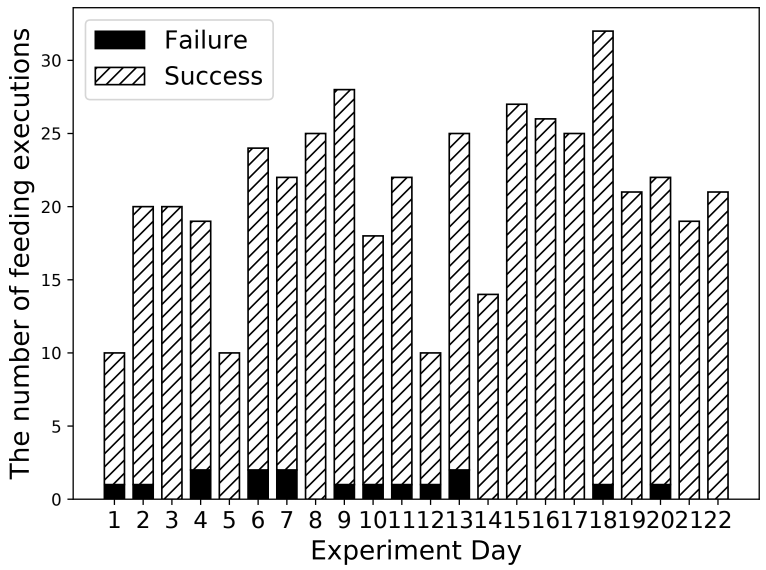


Figure 5.14: The number of daily successful and failed feeding executions during the long-term evaluation.

Remote Evaluation Before performing in-home evaluation, we first performed a remote test with the proto-type robot-assisted feeding system. From his home in California, USA, he used the web-based GUI to command a PR2 robot in Georgia, USA to feed an able-bodied person. Henry used an off-the-shelf head tracker and a mouse button to operate the web-based GUI. While using the system, he had visual feedback from the web-based GUI and a Beam+ (a separate telepresence robot). Figure 5.15 shows a capture of the remote test. Henry successfully used the system to feed yogurt to the able-bodied participant. When applicable, Henry answered survey questions. His answers, shown in Table 5.2, were similar to the able-bodied participants'. In response to an open-ended question at the end of the survey, he wrote "overall, worked well, although the PR2 video did not work." In a later email with the subject line "feeding feedback", he wrote "it is ready for field testing!", indicating he is prepared to try out the system in person.

In-home Evaluation We then performed an in-home evaluation with Henry Evans (see Figure 5.16 **Left**). He performed over 130 feeding executions during three-day evaluation at his home in California. He used the feeding system for two sessions per day, from

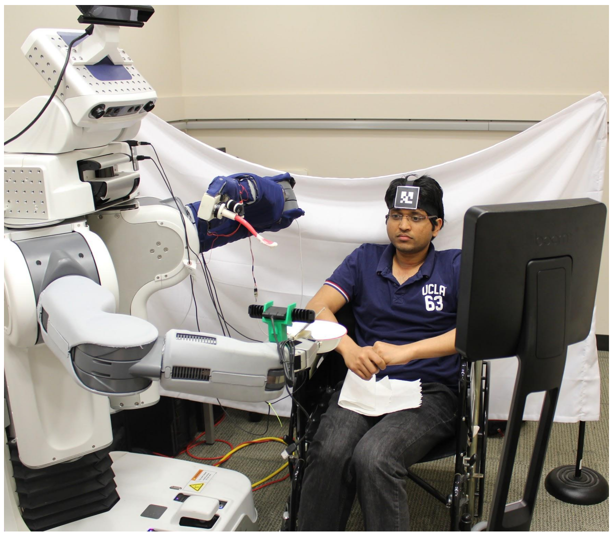


Figure 5.15: Our system feeding yogurt to an able-bodied person. A person with quadriplegia commanded the system remotely.

Table 5.2: Five-point Likert type questionnaire items for the remote test with Henry Evans. The last column provides answers with strongly disagree (sd), disagree (d), neither (n), agree (a), and strongly agree (sa).

Number	Question	Answer
1	The system was easy and intuitive to use.	sa
2	The web interface layout and icons were intuitive.	sa
3	I was satisfied with the time it took to complete the task.	d

February 11th through February 13th, 2017. For this real-world evaluation, we sent another PR2 robot from Georgia Tech to his home and mounted the same equipments as we used in laboratory. We used yogurt produced by Chobani, LLC and a silicone spoon. We conducted 1 session of system evaluation and 5 sessions of data collection for the execution monitoring system. This section will address the first session only.

We designed Henry Evans feeds himself without intervention from experimenters or caregivers. The experimenters located the robot holding a bowl and a spoon beside of his wheelchair on which Henry sat before this study. He used a head tracker to move the mouse cursor and an one-button mouse to click it (see Figure 5.16 **Right**). By manipulating the buttons on our GUI, he successfully controlled the system. We asked him to freely eat 20

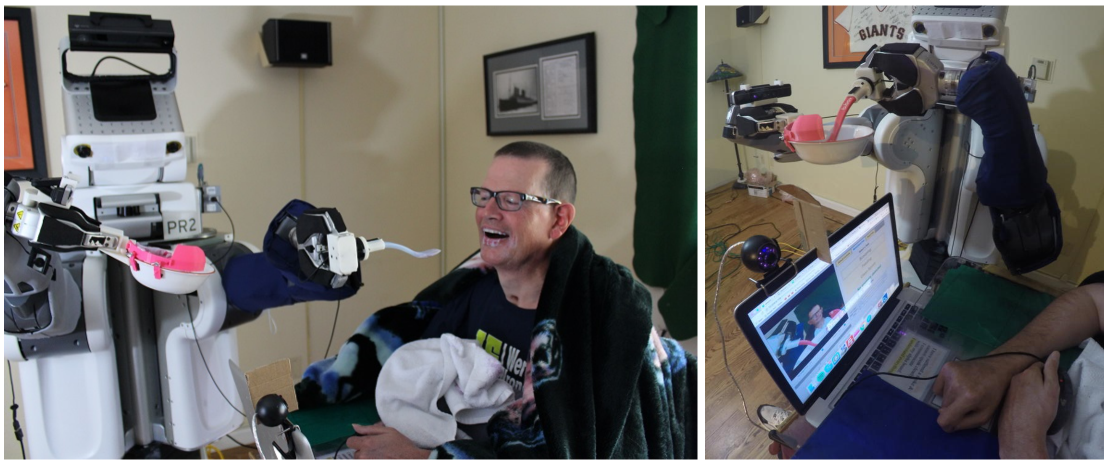


Figure 5.16: **Left** Henry Evans, a person with severe quadriplegia, successfully used our robot-assisted feeding system in his home to feed himself while our execution monitor was running. **Right** A capture of Henry's head tracker and one-button mouse.

scoops of yogurt using the Scooping, Wiping, and Feeding buttons. He successfully fed himself with the robot for all 20 consecutive trials. Note that we did not run the execution monitor to evaluate the feeding executions only.

At the end of our evaluation, we asked Henry to fill out a survey with 22 questions (five-point Likert type questionnaire items) based on [182], and 2 open-ended questions. Table 5.3 provides the questionnaire results that we found most informative. As can be seen from the table, Henry reported that he found the system to be effective, safe, and easy to use. His responses from the 22 questions indicated that the anomaly detection function positively contributed to his experience of using the robot, helping him feel safer, and effectively alerting him of problems. In an email following the experiment, Henry also recommended several ways to improve the system, such as increasing the rate at which it feeds yogurt and giving the user the ability to finely adjust where the spoon moves with respect to the mouth.

In-lab Evaluation We invited 3 participants who have difficulties in eating using hand-held utensils. Participants are male over 18 years of age. They participated in the study on their power wheelchair and fed themselves through the tablet able-bodied participants

Table 5.3: Five-point Likert type questionnaire items for an in-home evaluation with Henry Evans. The last column provides answers for strongly disagree (sd), disagree (d), neither (n), agree (a), and strongly agree (sa).

Number	Question	answer
1	The system successfully accomplished tasks.	sa
2	I felt safe while using the system.	sa
3	The system was simple and easy to use.	sa
4	The anomaly detection helped me feel more safe.	a

used. We performed 1 session of experiment lasting approximately 2 hours. After safety training and 3 times of practice trials, the participants were asked to freely perform 10 times of yogurt (or mixed fruit) feeding and another 10 times of mixed fruit (or yogurt) feeding. Experimenters refilled food after every 5 executions. The system successfully fed them 59 times out of 60 feeding attempts. The only failure was due to a participant's accidental stop button click. Note that we ran the execution monitor with an empirically determined low sensitivity but did not enable the monitor to stop the system to avoid any effects from false alarms during this feeding evaluation.

At the end of the experiment, we asked to answer questionnaires based on NASA TLX scores and five-point Likert type questionnaire items as well as 2 open-ended questions.

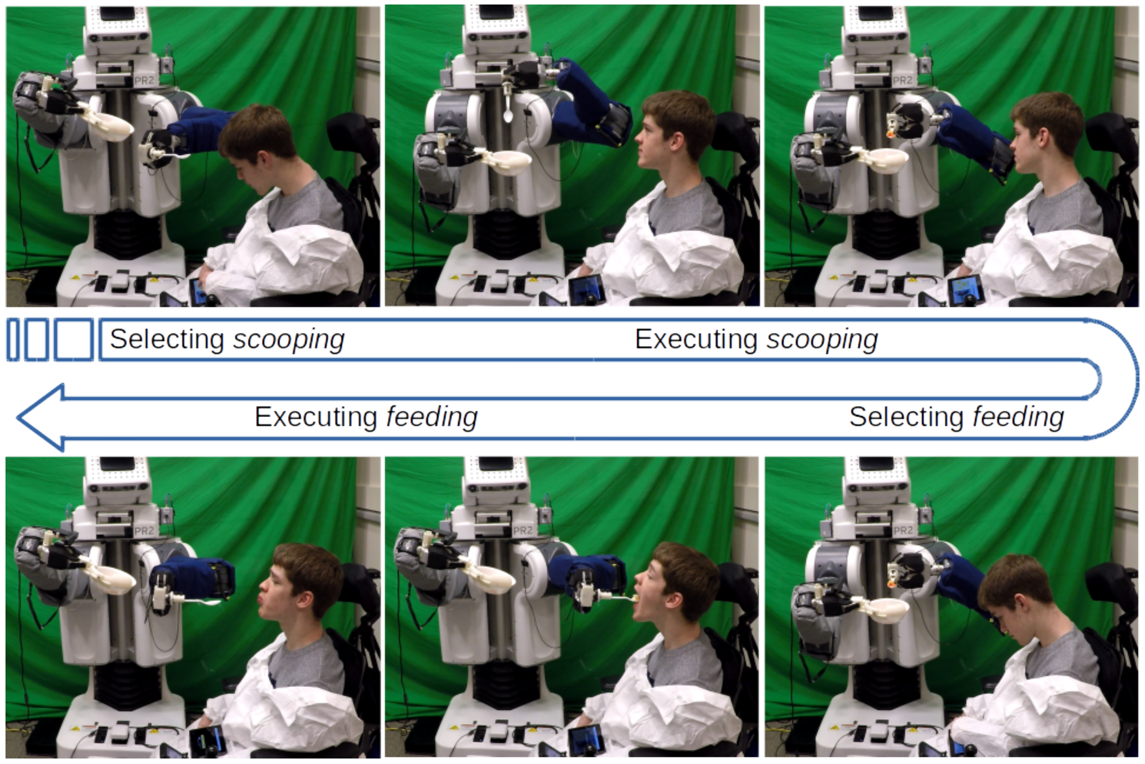


Figure 5.17: Evaluation with a person with disabilities in an experiment room in Georgia Tech.

5.4.3 Analyses

Completion Time Figure 5.18 shows the distributions of feeding completion time from 3 able-bodied participants and 3 people with disabilities. In this graph, each participant ate yogurt using a silicone spoon. The completion time is an elapsed time that has passed between the click of the scooping button and the end of feeding motion including spoon wiping executed by individuals' preference. Able-bodied participants and those of disabilities took about 39s and 44s, respectively. We performed Welch's t-test, also known as unequal variances t-test, that is to test two samples have equal means. The test resulted in p-*value* $= 0.0004$ that the two participant groups' completion times are significantly different. A likely cause is the difficulty of GUI manipulation due to upper-limb disabilities.

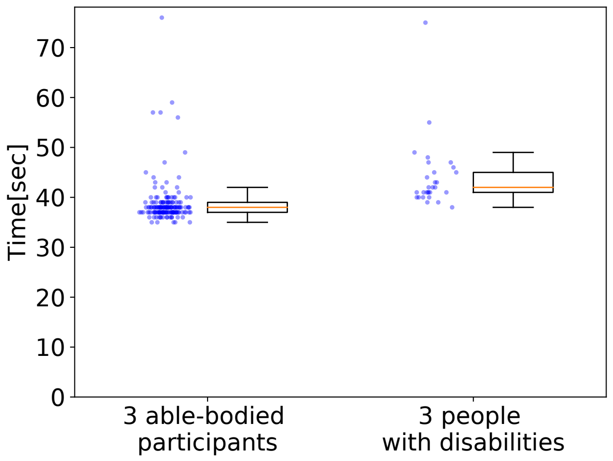


Figure 5.18: Distributions of robot-assisted feeding durations from 3 able-bodied participants and 3 people with disabilities (p-value $= 0.0004$ with a two-sided Welch's t-test). The participants ate yogurt with a silicone spoon.

Henry Evans also took about 78 seconds for one time of feeding that is about 39 seconds longer than the able-bodied participants' duration. A likely cause was from mouse pointing time using the head tracker and the use of wiping task where Henry Evans used it 0.85 times per feeding but the other participants mostly did not use it. Note that the wiping usually takes 17 seconds.

Ease of Use Our hypothesis is that 'the system is easy-to-use to performing self-feeding task.' To prove this hypothesis, we asked a question about ease of use to both able-bodied people and those with disabilities. Figure 5.19 shows their responses where the median responses are 'agree' for able-bodied people and 'strongly agree' for those with disabilities. The Welchs t-test resulted in $p\text{-}value = 0.2$ that indicates two groups have equal means.

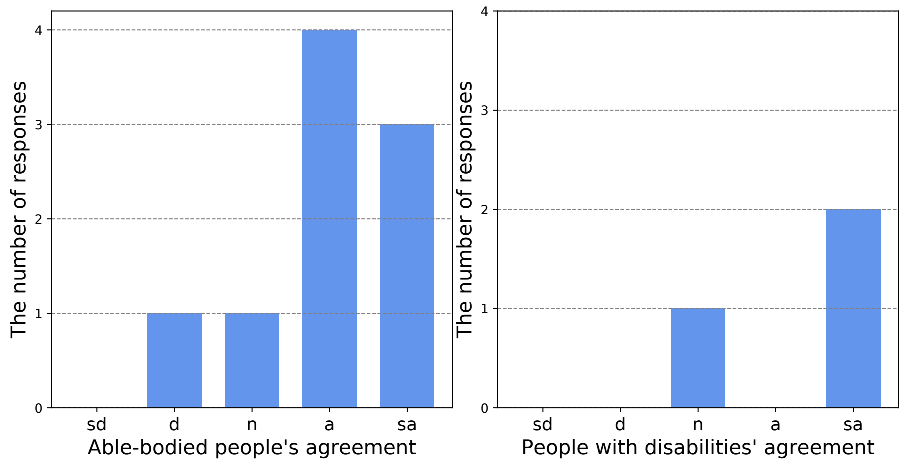


Figure 5.19: 9 able-bodied participants (left) and 3 people with disabilities (right) reported the level of agreement with the statement 'the system is easy-to-use to performing self-feeding task.' The two-sided Welch's t-test resulted in p-value $= 0.2$.

Comfortable Feeding Our hypothesis is that 'the system is comfortable to performing self-feeding task.' Figure 5.20 shows the participants' responses where the median responses are 'agree' for able-bodied people and 'strongly agree' for those with disabilities. The Welchs t-test resulted in $p\text{-}value = 0.28$ that indicates two groups have equal means.

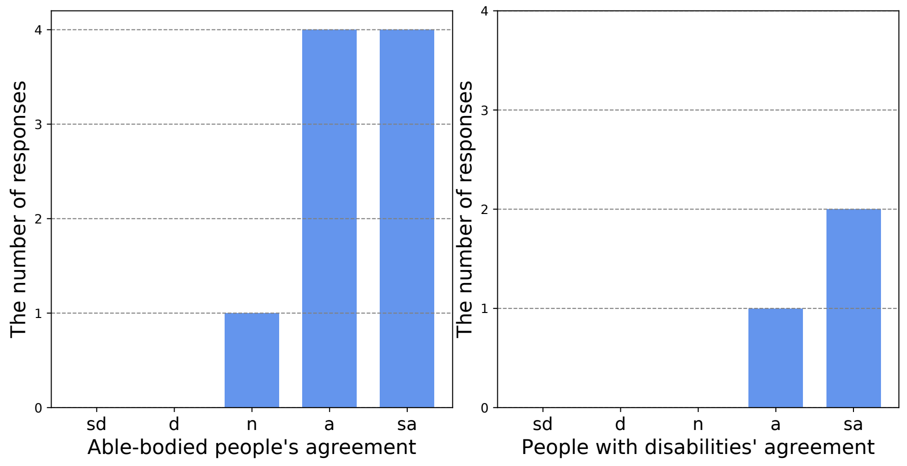


Figure 5.20: 9 able-bodied participants (**Left**) and 3 people with disabilities (**Right**) reported the level of agreement with the statement 'the system is comfortable to performing self-feeding task.' The two-sided Welch's t-test resulted in p-value $= 0.28$.

Successful Feeding Our hypothesis is that 'the system is successful to performing self-feeding task.' Figure 5.21 shows the participants' responses where the median responses are ‘agree’ for both. The Welchs t-test resulted in $p\text{-}value = 0.2$ that indicates two groups have equal means.

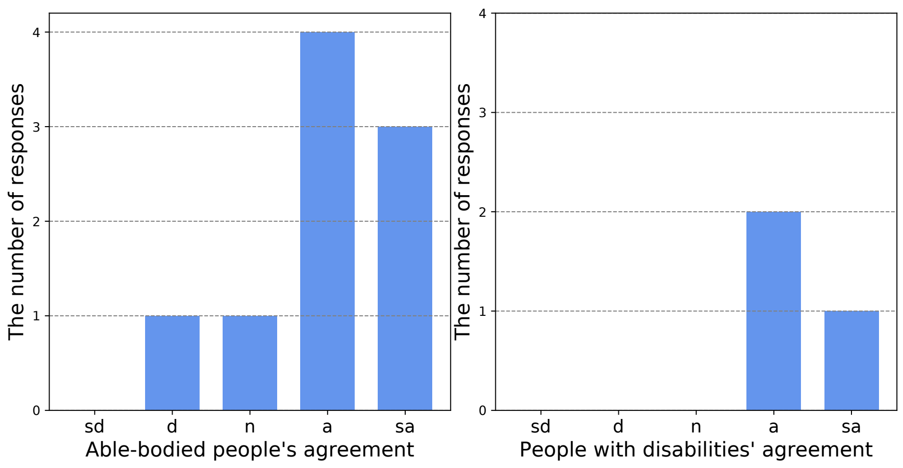


Figure 5.21: 9 able-bodied participants (**Left**) and 3 people with disabilities (**Right**) reported the level of agreement with the statement ‘the system is successful to performing self-feeding task.’ The two-sided Welch’s t-test resulted in p-value $= 0.2$.

Safety Our hypothesis is that ‘the system is safe to performing self-feeding task.’ Figure 5.22 shows the participants’ responses where the median responses are ‘strongly agree’ for both. The Welchs t-test resulted in $p\text{-}value = 0.44$ that indicates two groups have equal means.

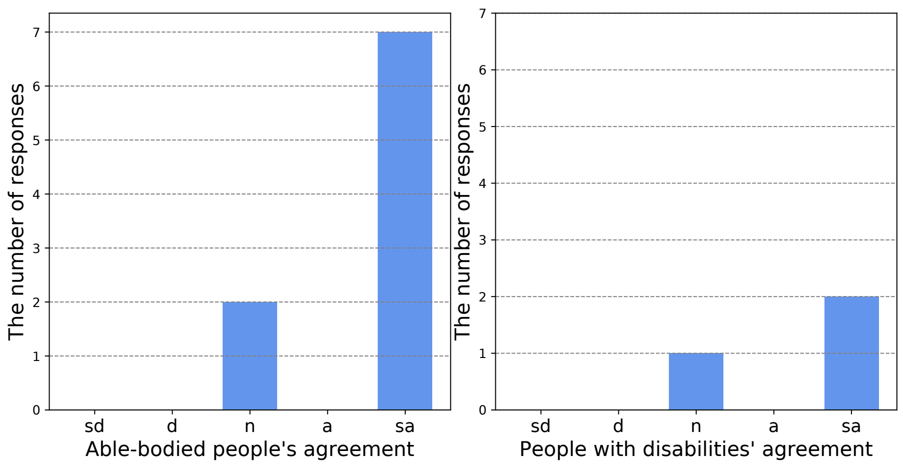


Figure 5.22: 9 able-bodied participants (**Left**) and 3 people with disabilities (**Right**) reported the level of agreement with the statement ‘the system is successful to performing self-feeding task.’ The two-sided Welch’s t-test resulted in p-value $= 0.44$.

Throughout the evaluations, our robot-assisted feeding system successfully fed foods to both able-bodied participants and those with disabilities. Participants agreed that the system comfortably, successfully, and safely provides the feeding assistance with easy-to-use interface. Overall, our results suggest that it is feasible for general-purpose mobile manipulators to provide feeding assistance.

CHAPTER 6
CONCLUSION

Assistance with activities of daily living (ADLs), such as feeding and toileting, are important for the quality of life. Yet for many people with disabilities, including the people with upper limb disabilities or quadriplegia, such tasks prove challenging without assistance from a human caregiver. However, the shortage of caregivers and high healthcare costs compared to care-receivers income create a pressing need for innovations that make assistance more affordable and effective. Assistive technology, such as feeding robots, can be a solution but each assistance device cost varies and typically provides a narrow form of assistance suitable for people with particular impairments

General-purpose mobile manipulators are proof-of-concept platforms that have the potential to serve as a versatile form of assistive technology. However, their complexity creates challenges, including the risk of being too difficult to use. In addition, their structural complexity, task variability, and sensor uncertainty may make these robots fail during assistance. These failures can be connected to hazards since the human body is easily vulnerable to physical contact.

To solve the above issues, we endowed a general-purpose mobile manipulator with the ability to provide safe assistance to diverse users for ADLs. Our solution is the development of an execution monitoring system using multimodal sensing. As a testbed, we focused on a representative ADL task—feeding—that is an essential ADL for the sustenance of good health but people with upper-limb disabilities, including individuals with quadriplegia, have difficulties feeding themselves. We developed a robot-assisted feeding system and an execution monitoring system that enables a PR2 robot from Willow Garage to assist people with disabilities with safe, easy-to-use task executions. The two systems' aims follow: 1) enable robots to recognize and regulate anomalous assistances caused by user, robot, and environment during the assistive tasks for safety; 2) enable conventional general-purpose mobile manipulators to provide autonomous assistance regardless of care receivers' physical limitations.

6.1 Contributions of Individual Chapters

6.1.1 Multimodal Anomaly Detection

We presented an execution monitoring system for multimodal anomaly detection during assistive robot manipulation. Our system modeled multimodal sensory readings associated with non-anomalous executions with a multivariate Gaussian HMM. Our approach takes advantage of similar patterns of change that tend to occur in hidden-state distributions and log-likelihoods during non-anomalous executions. Our system detects an anomaly when a log-likelihood is lower than a varying rejection threshold. Our methods varied the threshold with respect to an estimated execution progress. To estimate the threshold parameters, we introduced two methods, a clustering-based classifier (HMM-D) and a regression-based classifier (HMM-GP). We evaluated our methods with a PR2 robot performing object pushing and robot-assisted feeding tasks. Our methods outperformed 6 baseline methods from the literature by providing higher AUCs and shorter detection delays. We also showed that the detection of each anomaly tends to depend on a distinct set of modalities and that the use of multiple modalities was beneficial for anomaly detection.

We introduced an LSTM-VAE-based anomaly detector for multimodal anomaly detection. An LSTM-VAE models the underlying distribution of multi-dimensional signals and reconstructs the signals with expected distribution information. The detector estimated the negative log-likelihood of multimodal input with respect to the distribution as anomaly score. By introducing a denoising autoencoding criterion and state-based thresholding, the detector successfully detected anomalies in robot-assisted feeding resulting in higher AUC than 5 other baseline methods in literature. Without significant effort of feature engineering, the detector with 17 raw inputs outperformed a detector trained with 4 hand-engineered features. Finally, we also showed an LSTM-VAE with the state-based decision boundary is beneficial for more sensitive anomaly detection with lower false alarms.

6.1.2 Multimodal Anomaly Classification

We introduced a multimodal execution monitor that classifies the type and cause of common anomalies using an artificial neural network. We implemented and evaluated our execution monitor in the context of robot-assisted feeding with a general-purpose mobile manipulator. In our evaluations, our monitor outperformed 6 baseline methods from the literature. It succeeded in detecting 12 common anomalies from 8 able-bodied participants with 83% accuracy and classifying the types and causes of the detected anomalies with 90% and 81% accuracies, respectively. We then performed an in-home evaluation with Henry Evans, a person with severe quadriplegia. With our system, Henry successfully fed himself while the monitor detected, classified the types, and classified the causes of anomalies with 86%, 90%, and 54% accuracy, respectively. We also found multimodal features beneficial for classifying the causes of anomalies.

6.1.3 Application: A Robot-Assisted Feeding System

We introduced a proof-of-concept of robotic feeding system using a PR2 robot. The PR2 consists of a mobile base and two arms and each holds a utensil or a bowl, respectively. The PR2 can perform three independent tasks: scooping (or stabbing), wiping, and feeding. A user can command a preferred task via a web-based graphical user interface (GUI). We have evaluated the proto-type system with 10 able-bodied participants with approval from the Georgia Tech Institutional Review Board (IRB). The system successfully performed roughly 2,000 feeding executions with 5 utensils (small/large plastic spoons, a silicone spoon, and plastic/metal forks) and 6 foods (yogurt, rice, fruit mix, watermelon chunks, cereal, and cottage cheese). We also conducted in-home evaluation at Henry Evans' house in California, USA. The system successfully fed him with 70 non-anomalous feeding executions for three days.

6.2 Challenges and Opportunities

- Some unforeseen errors may not be detectable or identifiable by our proposed multi-modal execution monitor if we do not input a set of modalities relevant to the errors. This may decrease the usability of both the monitoring and assistance system. The addition of new modality or incremental learning of new anomalies will be helpful to make the systems robust in real settings.

- The user, task, and environmental variabilities may lower the performance of the monitoring system in real settings. To keep its sensitivity and lower false alarms, the monitoring system also needs to adapt its internal parameters or learned models to new data from new users or environments. This is strongly related with reinforcement learning (RL) or iterative update (or optimization).

- In addition, new task and environment may require the extension of the monitoring system to detect and recognize new classes of anomalous executions. Particularly, supervised anomaly classifiers can be trained from scratch but this is computationally expensive. Instead, the system needs to extend its network structure by adding new output classes and partially update its parameter. This is relevant to network surgery and incremental learning.

- Another issue is the selection of recovery strategies. The current system provides opportunities to automatically correct the robots' behaviors by sending the detected and classified information to the robots. Current recovery plans are restricted to stopping, resuming, or initializing actions. By selecting diverse strategies, the system will be able to improve the effectiveness of the assistance. For example, after the detection and identification of a spoon-face collision, the system can brings paper towers and then wipes out the food on the face.

- The users may have different physical abilities and personalities. The environments they are in may largely vary. Due to the variance, the patterns and magnitudes of sensed signals may vary. These affect the performance of the monitoring system. For precise anomaly detection with lower false alarms, the monitoring system needs to be adapted to a current user and environment online.

- The cost of a general-purpose robot is decreasing rapidly due to growing commercial interest. In 2010, Willow Garage released a wheeled dual manipulator, the PR2, for \$400,000. Only two years later, Rethink Robotics released the noticeably dropped cost of a dual manipulator, the Baxter robot, for \$22,000, while the Obi feeding robot, a single-purpose robot, costs \$5,950. In addition, open-source software and hardware have been rapidly improved in availability and quality over the last decade. We believe the development of assistive applications using general-purpose mobile manipulators could be economically affordable in the near future. It will then be a solution for the aging population, rising healthcare costs, and shortage of healthcare workers in the United States. We also expect the inventions have the potential to reduce family caregivers' prolonged stress, physical demands, and decline in quality of life while reducing the healthcare cost for households with people with disabilities.

- The system is not for rehabilitation but for assistance. The automated assistance enables people with diverse disabilities to perform daily activities by themselves. However, it may discourage users with light disabilities from using their physical abilities, though they can perform the activities on some level. It may also affect their rehabilitation status and worsen their health. We believe future assistive robots will increase rehabilitation opportunities and reduce users' excessive reliance on the technology.

These contributions and discussions constitute a first step towards our longer-term goal of a new foundation for assistive robots. We would like to enable assistive robots to provide safe and reliable assistance without caregivers and consequently provide the equalization of opportunities for people with disabilities.

Appendices

APPENDIX A
CODE, VIDEO, AND DATA RELEASE

A.1 Code

The code for the execution monitoring and the robot-assisted feeding has been made available through following public repositories:

- **Execution Monitor**: `https://github.com/gt-ros-pkg/hrl-assistive/tree/indigo-devel/hrl_execution_monitor`
- **Robot-Assisted Feeding**: `https://github.com/gt-ros-pkg/hrl-assistive/tree/indigo-devel/hrl_manipulation_task`

A.2 Video

Videos for the execution monitoring and the robot-assisted feeding are available at:

- Multimodal anomaly detection with HMM-D: `https://youtu.be/gLcPZQnDmkk`
- Multimodal anomaly detection with LSTM-VAE: `https://youtu.be/ZMAGEQx5Uy8`
- Multimodal anomaly classification: `https://youtu.be/KQlVSz3URnA`

A.3 Data

The training and testing data for the multimodal execution monitoring has been also made available through following file server, `ftp://ftp-hrl.bme.gatech.edu`.

REFERENCES

[1] J. M. Wiener, R. J. Hanley, R. Clark, and J. F. Van Nostrand, “Measuring the activities of daily living: Comparisons across national surveys,” *Journal of Gerontology*, vol. 45, no. 6, S229–S237, 1990.

[2] K. P. Hawkins, P. M. Grice, T. L. Chen, C.-H. King, and C. C. Kemp, “Assistive mobile manipulation for self-care tasks around the head,” in *IEEE Symposium on Computational Intelligence in Robotic Rehabilitation and Assistive Technologies*, IEEE, 2014, pp. 16–25.

[3] Y. Wakita, W. K. Yoon, and N. Yamanobe, “User evaluation to apply the robotic arm rapuda for an upper-limb disabilities patient’s daily life,” in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 1482–1487.

[4] C. H. King, T. L. Chen, A. Jain, and C. C. Kemp, “Towards an assistive robot that autonomously performs bed baths for patient hygiene,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 319–324.

[5] S. Schrer, I. Killmann, B. Frank, M. Vlker, L. Fiederer, T. Ball, and W. Burgard, “An autonomous robotic assistant for drinking,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6482–6487.

[6] C. Copilusi, M. Kaur, and M. Ceccarelli, “Lab experiences with larm clutched arm for assisting disabled people,” in *New Trends in Mechanism and Machine Science: From Fundamentals to Industrial Applications*, P. Flores and F. Viadero, Eds. Cham: Springer International Publishing, 2015, pp. 603–611, ISBN: 978-3-319-09411-3.

[7] Y. Takahashi and S. Suzukawa, “Easy human interface for severely handicapped persons and application for eating assist robot,” in *2006 IEEE International Conference on Mechatronics*, Jul. 2006, pp. 225–229.

[8] D. Park, Y. K. Kim, Z. Erickson, and C. C. Kemp, “Towards assistive feeding with a general-purpose mobile manipulator,” in *IEEE International Conference on Robotics and Automation - workshop on Human-Robot Interfaces for Enhanced Physical Interactions*, 2016.

[9] J. Silvrio, L. Rozo, S. Calinon, and D. G. Caldwell, “Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sep. 2015, pp. 464–470.

[10] D. Leidner, A. Dietrich, M. Beetz, and A. Albu-Schäffer, “Knowledge-enabled parameterization of whole-body control strategies for compliant service robots,” *Autonomous Robots*, vol. 40, no. 3, pp. 519–536, 2016.

[11] O. Pettersson, “Execution monitoring in robotics: A survey,” *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005.

[12] C. C. Kemp, A. Edsinger, and E. Torres-Jara, “Challenges for robot manipulation in human environments [grand challenges of robotics],” *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 20–29, 2007.

[13] A. Bouguerra, L. Karlsson, and A. Saffiotti, “Monitoring the execution of robot plans using semantic knowledge,” *Robotics and autonomous systems*, vol. 56, no. 11, pp. 942–954, 2008.

[14] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[15] Y. Zhang, N. Meratnia, and P. Havinga, “Outlier detection techniques for wireless sensor networks: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 159–170, 2010.

[16] O. Safety, H. Administration, *et al.*, *Osha technical manual (otm), section iv: Chapter 4. industrial robots and robot system safety*, U.S. Department of Labor.

[17] O. Ogorodnikova, “Methodology of safety for a human robot interaction designing stage,” in *2008 Conference on Human System Interactions*, IEEE, 2008, pp. 452–457.

[18] M. Vasic and A. Billard, “Safety issues in human-robot interactions,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 197–204.

[19] M. Bjäreland, “Model-based execution monitoring,” in *Linköping Studies in Science and Technology, Dissertation No 688*, Citeseer, 2001.

[20] B. S. Dhillon, *Robot reliability and safety*. Springer Science & Business Media, 2012.

[21] B. Plattner and J. Nievergelt, “Special feature: Monitoring program execution: A survey,” *Computer*, vol. 14, no. 11, pp. 76–93, 1981.

[22] E. R. Sanseverino, M. L. Di Silvestre, M. G. Ippolito, A. De Paola, and G. L. Re, “An execution, monitoring and replanning approach for optimal energy management in microgrids,” *Energy*, vol. 36, no. 5, pp. 3429–3436, 2011.

[23] M. De Leoni, M. Mecella, and G. De Giacomo, “Highly dynamic adaptation in process management systems through execution monitoring,” in *International Conference on Business Process Management*, Springer, 2007, pp. 182–197.

[24] E. Gat, M. G. Slack, D. P. Miller, and R. J. Firby, “Path planning and execution monitoring for a planetary rover,” in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, IEEE, 1990, pp. 20–25.

[25] F. R. Noreils and R. G. Chatila, “Plan execution monitoring and control architecture for mobile robots,” *IEEE transactions on robotics and automation*, vol. 11, no. 2, pp. 255–266, 1995.

[26] J. L. Fernandez and R. G. Simmons, “Robust execution monitoring for navigation plans,” in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, IEEE, vol. 1, 1998, pp. 551–557.

[27] K. Z. Haigh and M. M. Veloso, “Interleaving planning and robot execution for asynchronous user requests,” in *Autonomous agents*, Springer, 1998, pp. 79–95.

[28] A. Jain and C. C. Kemp, “El-e: An assistive mobile manipulator that autonomously fetches objects from flat surfaces,” *Autonomous Robots*, vol. 28, no. 1, pp. 45–64, 2010.

[29] ——, “Improving robot manipulation with data-driven object-centric models of everyday forces,” *Autonomous Robots*, vol. 35, no. 2-3, pp. 143–159, 2013.

[30] T. Sakaguchi, K. Yokoi, T. Ujiie, S. Tsunoo, and K. Wada, “Design of common environmental information for door-closing tasks with various robots,” *International Journal of Robotics & Automation*, vol. 24, no. 3, p. 203, 2009.

[31] M. Topping, “An overview of the development of handy 1, a rehabilitation robot to assist the severely disabled,” *Journal of intelligent and robotic systems*, vol. 34, no. 3, pp. 253–263, 2002.

[32] K Alho, T Kujala, P Paavilainen, H Summala, and R Näätänen, “Auditory processing in visual brain areas of the early blind: Evidence from event-related potentials,” *Electroencephalography and clinical neurophysiology*, vol. 86, no. 6, pp. 418–427, 1993.

[33] P. Fitzpatrick, A. Arsenio, and E. R. Torres-Jara, “Reinforcing robot perception of multi-modal events through repetition and redundancy and repetition and redundancy,” *Interaction Studies*, vol. 7, no. 2, pp. 171–196, 2006.

[34] H. Wu and M. Siegel, "Correlation of accelerometer and microphone data in the coin tap testi," *Instrumentation and Measurement, IEEE Transactions on*, vol. 49, no. 3, pp. 493–497, 2000.

[35] Z. Su, O. Kroemer, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Learning to switch between sensorimotor primitives using multimodal haptic signals," in *From Animals to Animats 14: 14th International Conference on Simulation of Adaptive Behavior, SAB 2016, Aberystwyth, UK, August 23-26, 2016, Proceedings*, E. Tuci, A. Giagkos, M. Wilson, and J. Hallam, Eds. Cham: Springer International Publishing, 2016, pp. 170–182, ISBN: 978-3-319-43488-9.

[36] J. Wade, T. Bhattacharjee, R. D. Williams, and C. C. Kemp, "A force and thermal sensing skin for robots in human environments," *Robotics and Autonomous Systems*, vol. 96, pp. 1 –14, 2017.

[37] A. Rodriguez, D. Bourne, M. Mason, G. F. Rossano, and J. Wang, "Failure detection in assembly: Force signature analysis," in *Automation Science and Engineering (CASE), IEEE Conference on*, IEEE, 2010, pp. 210–215.

[38] S. Ando, E. Suzuki, Y. Seki, T. Thanongphongphan, and D. Hoshino, "Ace: Anomaly clustering ensemble for multi-perspective anomaly detection in robot behaviors.," in *SDM*, SIAM, 2011, pp. 1–12.

[39] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *Robotics and Automation (ICRA), IEEE International Conference on*, IEEE, 2011, pp. 3828–3834.

[40] V. Chu, I. McMahon, L. Riano, C. G. McDonald, Q. He, J. Martinez Perez-Tejada, M. Arrigo, N. Fitter, J. C. Nappo, T. Darrell, *et al.*, "Using robotic exploratory procedures to learn the meaning of haptic adjectives," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 3048–3055.

[41] D. A. Clifton, S. Hugueny, and L. Tarassenko, "Novelty detection with multivariate extreme value statistics," *Journal of signal processing systems*, vol. 65, no. 3, pp. 371–389, 2011.

[42] D. Park, Z. Erickson, T. Bhattacharjee, and C. C. Kemp, "Multimodal execution monitoring for anomaly detection during robot manipulation," in *Robotics and Automation, 2016. ICRA'16. IEEE International Conference on*, IEEE, 2016.

[43] O. Pettersson, L. Karlsson, and A. Saffiotti, *Model-free execution monitoring in behavior-based mobile robotics*. Örebro universitetsbibliotek, 2004.

[44] A. Haidu, D. Kohlsdorf, and M. Beetz, "Learning action failure models from interactive physics-based simulations," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 5370–5375.

[45] D. Kappler, P. Pastor, M. Kalakrishnan, M. Wuthrich, and S. Schaal, "Data-driven online decision making for autonomous manipulation," in *Proceedings of Robotics: Science and Systems*, Jul. 2015.

[46] J. Simanek, V. Kubelka, and M. Reinstein, "Improving multi-modal data fusion by anomaly detection," *Autonomous Robots*, vol. 39, no. 2, pp. 139–154, 2015.

[47] M. Markou and S. Singh, "Novelty detection: A reviewpart 1: Statistical approaches," *Signal processing*, vol. 83, no. 12, pp. 2481–2497, 2003.

[48] S. Dua and X. Du, *Data mining and machine learning in cybersecurity*. CRC press, 2011.

[49] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[50] A. C. Bittencourt, K. Saarinen, S. Sander-Tavallaey, S. Gunnarsson, and M. Norrlöf, "A data-driven approach to diagnostics of repetitive processes in the distribution domain–applications to gearbox diagnostics in industrial robots and rotating machines," *Mechatronics*, 2014.

[51] D. Brambilla, L. M. Capisani, A. Ferrara, and P. Pisu, "Fault detection for robot manipulators via second-order sliding modes," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 11, pp. 3954–3963, Nov. 2008.

[52] S. Blank, T. Pfister, and K. Berns, "Sensor failure detection capabilities in low-level fusion: A comparison between fuzzy voting and kalman filtering," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 4974–4979.

[53] A. Suarez, G. Heredia, and A. Ollero, "Cooperative sensor fault recovery in multi-uav systems," in *Robotics and Automation, 2016. ICRA'16. IEEE International Conference on*, IEEE, 2016.

[54] N. F. Lepora, M. J. Pearson, B. Mitchinson, M. Evans, C. Fox, A. Pipe, K. Gurney, and T. J. Prescott, "Naive bayes novelty detection for a moving robot with whiskers," in *Robotics and Biomimetics (ROBIO), IEEE International Conference on*, IEEE, 2010, pp. 131–136.

[55] P. M. Dames, M. Schwager, D. Rus, and V. Kumar, “Active magnetic anomaly detection using multiple micro aerial vehicles,” *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 153–160, Jan. 2016.

[56] H. Suetani, A. M. Ideta, and J. Morimoto, “Nonlinear structure of escape-times to falls for a passive dynamic walker on an irregular slope: Anomaly detection using multi-class support vector machine and latent state extraction by canonical correlation analysis,” in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, IEEE, 2011, pp. 2715–2722.

[57] F. Marcolino and J. Wang, “Detecting anomalies in humanoid joint trajectories,” in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, IEEE, 2013, pp. 2594–2599.

[58] A. Rodriguez, M. T. Mason, S. Srinivasa, M. Bernstein, and A. Zirbel, “Abort and retry in grasping,” in *IEEE International Conference on Intelligent Robots and Systems (IROS 2011)*, Sep. 2011.

[59] V. Sukhoy, V. Georgiev, T. Wegter, R. Sweidan, and A. Stoytchev, “Learning to slide a magnetic card through a card reader,” in *Robotics and Automation (ICRA), IEEE International Conference on*, IEEE, 2012, pp. 2398–2404.

[60] M. Geravand, W. Rampeltshammer, and A. Peer, “Control of mobility assistive robot for human fall prevention,” in *2015 IEEE International Conference on Rehabilitation Robotics (ICORR)*, Aug. 2015.

[61] A. Colombo, D. Fontanelli, A. Legay, L. Palopoli, and S. Sedwards, “Efficient customisable dynamic motion planning for assistive robots in complex human environments,” *Journal of ambient intelligence and smart environments*, vol. 7, no. 5, pp. 617–634, 2015.

[62] R. Hornung, H. Urbanek, J. Klodmann, C. Osendorfer, and P. van der Smagt, “Model-free robot anomaly detection,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3676–3683.

[63] K. Häussermann, O. Zweigle, and P. Levi, “A novel framework for anomaly detection of robot behaviors,” *Journal of Intelligent & Robotic Systems*, vol. 77, no. 2, pp. 361–375, 2015.

[64] S. Luo, H. Wu, H. Lin, S. Duan, Y. Guan, and J. Rojas, “Robust and versatile event detection through gradient-based scoring of HMM models,” *ArXiv preprint arXiv:1709.07876*, 2017.

[65] J.-i. Furukawa, T. Noda, T. Teramae, and J. Morimoto, “Estimating joint movements from observed emg signals with multiple electrodes under sensor failure

situations toward safe assistive robot control," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 4985–4991.

[66] H. Fujii, A. Yamashita, and H. Asama, "Defect detection with estimation of material condition using ensemble learning for hammering test," in *Robotics and Automation, 2016. ICRA'16. IEEE International Conference on*, IEEE, 2016.

[67] S. Niekum, S. Osentoski, C. G. Atkeson, and A. G. Barto, "Learning articulation changepoint models from demonstration," in *Robotics: Science and Systems (RSS) Workshop on Learning Plans with Context from Human Signals*, 2014.

[68] G. Fagogenis, V. D. Carolis, and D. M. Lane, "Online fault detection and model adaptation for underwater vehicles in the case of thruster failures," in *Robotics and Automation, 2016. ICRA'16. IEEE International Conference on*, IEEE, 2016.

[69] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, IEEE, 1999, pp. 133–145.

[70] V. Perduca and G. Nuel, "Measuring the influence of observations in HMMs through the kullback–leibler distance," *IEEE Signal Processing Letters*, vol. 20, no. 2, pp. 145–148, 2013.

[71] B. T. Morris and M. M. Trivedi, "Learning and classification of trajectories in dynamic scenes: A general framework for live video analysis," in *Advanced Video and Signal Based Surveillance, IEEE Fifth International Conference on*, IEEE, 2008, pp. 154–161.

[72] J. P. Mendoza, M. Veloso, and R. Simmons, "Focused optimization for online detection of anomalous regions," in *Robotics and Automation (ICRA), IEEE International Conference on*, IEEE, 2014, pp. 3358–3363.

[73] C. Song, K. Liu, and X. Zhang, "Integration of data-level fusion model and kernel methods for degradation modeling and prognostic analysis," *IEEE Transactions on Reliability*, vol. PP, no. 99, pp. 1–11, 2017.

[74] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[75] K. Noda, H. Arie, Y. Suga, and T. Ogata, "Multimodal integration learning of robot behavior using deep neural networks," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 721–736, 2014.

[76] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[77] T. J. O'Shea, T. C. Clancy, and R. W. McGwier, "Recurrent neural radio anomaly detection," *ArXiv preprint arXiv:1611.00301*, 2016.

[78] S. Chauhan and L. Vig, "Anomaly detection in ecg time signals via deep long short-term memory networks," in *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, IEEE, 2015, pp. 1–7.

[79] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2015.

[80] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," in *Anomaly Detection Workshop at 33rd International Conference on Machine Learning (ICML 2016)*, CoRR, abs/1607.00148, 2016, 2016.

[81] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.

[82] E. Principi, F. Vesperini, S. Squartini, and F. Piazza, "Acoustic novelty detection with adversarial autoencoders," in *Neural Networks (IJCNN), 2017 International Joint Conference on*, IEEE, 2017, pp. 3324–3330.

[83] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the Second International Conference on Learning Representations (ICLR)*, Apr. 2014.

[84] J. Bayer and C. Osendorfer, "Learning stochastic recurrent networks," in *NIPS 2014 Workshop on Advances in Variational Inference*, 2014.

[85] M. Sölch, J. Bayer, M. Ludersdorfer, and P. van der Smagt, "Variational inference for on-line anomaly detection in high-dimensional time series," *ICML 2016 Anomaly Detection Workshop*, 2016.

[86] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 10–21.

[87] R. Isermann, "Supervision, fault-detection and fault-diagnosis methodsan introduction," *Control engineering practice*, vol. 5, no. 5, 1997.

[88] F. Caccavale and L. Villani, *Fault diagnosis and fault tolerance for mechatronic systems: Recent advances*. Springer Science & Business Media, 2002, vol. 1.

[89] R. Muradore and P. Fiorini, "A pls-based statistical approach for fault detection and isolation of robotic manipulators," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3167–3175, 2012.

[90] L. H. Chiang, R. D. Braatz, and E. L. Russell, *Fault detection and diagnosis in industrial systems*. Springer Science & Business Media, 2001.

[91] P. Jackson, "Introduction to expert systems," 1986.

[92] R. Tinós and M. H. Terra, "Fault detection and isolation in robotic manipulators using a multilayer perceptron and a rbf network trained by the kohonens self-organizing map," *Rev Soc Bras Autom Contr Autom*, vol. 12, no. 1, pp. 11–18, 2001.

[93] T. Yüksel and A. Sezgin, "Two fault detection and isolation schemes for robot manipulators using soft computing techniques," *Applied Soft Computing*, vol. 10, no. 1, pp. 125–134, 2010.

[94] K. Yamazaki, R. Oya, K. Nagahama, K. Okada, and M. Inaba, "Bottom dressing by a life-sized humanoid robot provided failure detection and recovery functions," in *2014 IEEE/SICE International Symposium on System Integration*, Dec. 2014, pp. 564–570.

[95] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[96] J. Sung, S. H. Jin, I. Lenz, and A. Saxena, "Robobarista: Learning to manipulate novel objects via deep multimodal embedding," *ArXiv preprint arXiv:1601.02705*, 2016.

[97] P. Angelov, V Giglio, C Guardiola, E. Lughofer, and J. Lujan, "An approach to model-based fault detection in industrial measurement systems with application to engine test benches," *Measurement Science and Technology*, vol. 17, no. 7, p. 1809, 2006.

[98] F. Serdio, E. Lughofer, K. Pichler, T. Buchegger, and H. Efendic, "Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills," *Information Sciences*, vol. 259, pp. 304–320, 2014.

[99] S. Lühr, S. Venkatesh, G. West, and H. H. Bui, "Explicit state duration HMM for abnormality detection in sequences of human activity," in *Pacific Rim International Conference on Artificial Intelligence*, Springer, 2004, pp. 983–984.

[100] S. S. Khan, M. E. Karg, J. Hoey, and D. Kulic, "Towards the detection of unusual temporal events during activities using HMMs," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ACM, 2012, pp. 1075–1084.

[101] N. Vaswani, A. K. Roy-Chowdhury, and R. Chellappa, ""shape activity": A continuous-state HMM for moving/deforming shapes with application to abnormal activity detection," *Image Processing, IEEE Transactions on*, vol. 14, no. 10, pp. 1603–1616, 2005.

[102] E. Di Lello, M. Klotzbucher, T. De Laet, and H. Bruyninckx, "Bayesian time-series models for continuous fault detection and recognition in industrial robotic tasks," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2013, pp. 5827–5833.

[103] H. Ocak and K. A. Loparo, "HMM-based fault detection and diagnosis scheme for rolling element bearings," *Journal of Vibration and Acoustics*, vol. 127, no. 4, pp. 299–306, 2005.

[104] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern recognition*, vol. 36, no. 1, pp. 229–243, 2003.

[105] D. Feil-Seifer and M. J. Mataric, "Defining socially assistive robotics," in *International Conference on Rehabilitation Robotics (ICORR)*, IEEE, 2005, pp. 465–468.

[106] P. Maciejasz, J. Eschweiler, K. Gerlach-Hahn, A. Jansen-Troy, and S. Leonhardt, "A survey on robotic devices for upper limb rehabilitation," *Journal of neuroengineering and rehabilitation*, vol. 11, no. 1, p. 3, 2014.

[107] S. Ishii, S. Tanaka, and F. Hiramatsu, "Meal assistance robot for severely handicapped people," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1995, pp. 1308–1313.

[108] W. Yu, A. Kapusta, J. Tan, C. C. Kemp, G. Turk, and C. K. Liu, "Haptic simulation for robot-assisted dressing," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 6044–6051.

[109] J. Hammel, K. Hall, D. Lees, L. Leifer, M. Van der Loos, I. Perkash, and R. Crigler, "Clinical evaluation of a desktop robotic assistant," *J Rehabil Res Dev*, vol. 26, no. 3, pp. 1–16, 1989.

[110] H. M. Van der Loos, J. J. Wagner, N. Smaby, K Chang, O. Madrigal, L. J. Leifer, and O. Khatib, "ProVAR assistive robot system architecture," in *IEEE International Conference on Robotics and Automation*, IEEE, vol. 1, 1999, pp. 741–746.

[111] M. Hillman, K. Hagan, S. Hagan, J. Jepson, and R. Orpwood, "A wheelchair mounted assistive robot," in *Proceedings of the RESNA'99 Annual Conference: Spotlight on Technology*, ERIC, 1999, p. 125.

[112] J. Dietsch, A Jardon, A Gimenez, R Correal, R Cabas, S Martinez, and C Balaguer, "A portable light-weight climbing robot for personal assistance applications," *Industrial Robot: An International Journal*, vol. 33, no. 4, pp. 303–307, 2006.

[113] Y. Wakita, W.-K. Yoon, and N. Yamanobe, "User evaluation to apply the robotic arm RAPUDA for an upper-limb disabilities patient's daily life," in *Robotics and Biomimetics (ROBIO), IEEE International Conference on*, IEEE, 2012, pp. 1482–1487.

[114] V. Maheu, P. S. Archambault, J. Frappier, and F. Routhier, "Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities," in *Rehabilitation Robotics (ICORR), IEEE International Conference on*, IEEE, 2011, pp. 1–5.

[115] S. Schroer, I. Killmann, B. Frank, M. Volker, L. Fiederer, T. Ball, and W. Burgard, "An autonomous robotic assistant for drinking," in *Robotics and Automation, IEEE International Conference on*, 2015.

[116] D.-J. Kim, Z. Wang, N. Paperno, and A. Behal, "System design and implementation of ucf-manusan intelligent assistive robotic manipulator," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 1, pp. 225–237, 2014.

[117] A. Campeau-Lecours, V. Maheu, S. Lepage, H. Lamontagne, S. Latour, L. Paquet, and N. Hardie, "Jaco assistive robotic device: Empowering people with disabilities through innovative algorithms," in *Rehabilitation Engineering and Assistive Technology Society of North America (RESNA) annual conference*, vol. 14, 2016.

[118] A. Kapusta, D. Park, and C. C. Kemp, "Task-centric selection of robot and environment initial configurations for assistive tasks," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2015, pp. 1480–1487.

[119] T. L. Chen, M. Ciocarlie, S. Cousins, P. M. Grice, K. Hawkins, K. Hsiao, C. C. Kemp, C.-H. King, D. A. Lazewatsky, A. E. Leeper, *et al.*, "Robots for humanity: Using assistive robotics to empower people with disabilities," *IEEE Robotics & Automation Magazine*, vol. 20, no. 1, pp. 30–39, 2013.

[120] M. Ciocarlie, K. Hsiao, A. Leeper, and D. Gossow, “Mobile manipulation through an assistive home robot,” in *Intelligent Robots and Systems , 2012 IEEE/RSJ International Conference on*, IEEE, 2012.

[121] B. Graf, U. Reiser, M. Hägele, K. Mauz, and P. Klein, “Robotic home assistant care-o-bot® 3-product vision and innovation platform,” in *Advanced Robotics and its Social Impacts (ARSO), 2009 IEEE Workshop on*, IEEE, 2009, pp. 139–144.

[122] H. Nguyen, C. Anderson, A. Trevor, A. Jain, Z. Xu, and C. C. Kemp, “El-e: An assistive robot that fetches objects from flat surfaces,” in *Robotic helpers, int. conf. on human-robot interaction*, 2008.

[123] A Kargov, T Asfour, C Pylatiuk, R Oberle, H Klosek, S Schulz, K Regenstein, G Bretthauer, and R Dillmann, “Development of an anthropomorphic hand for a mobile assistive robot,” in *Rehabilitation Robotics, 9th International Conference on*, IEEE, 2005, pp. 182–186.

[124] Neater Solutions, Neater Eaters, `http://www.neater.co.uk/` [Accessed: 2017-07-15], 2017.

[125] Liftware, Liftware, `https://www.liftware.com/` [Accessed: 2017-07-15], 2014.

[126] R. P. Hermann, A. C. Phalangas, R. M. Mahoney, and M. Alexander, “Powered feeding devices: An evaluation of three models,” *Archives of physical medicine and rehabilitation*, vol. 80, no. 10, pp. 1237–1242, 1999.

[127] Secom, Meal-Assistance Robot, My Spoon, `https://www.secom.co.jp/english/myspoon/` [Accessed: 2017-07-15], Secom, Inc., 2017.

[128] Camanio Care AB, Bestic, Increase your mealtime independence, `http://www.camanio.com/` [Accessed: 2017-07-15], 2017.

[129] Mealtime Partners, Specializing in Assistive Dining and Drinking Equipment, `http://www.mealtimepartners.com/` [Accessed: 2017-07-15], Mealtime Partners, Inc., 2017.

[130] Patterson Medical, *Meal buddy*, `http://pattersonmedical.com/` [Accessed: 2017-07-15], 2017.

[131] Eclipse Automation, *Meet obi, a robot that helps disabled individuals eat unassisted*, `https://meetobi.com/` [Accessed: 2017-07-15], 2016.

[132] I. Naotunna, C. J. Perera, C. Sandaruwan, R. Gopura, and T. D. Lalitharatne, “Meal assistance robots: A review on current status, challenges and future directions,”

in *IEEE/SICE International Symposium on System Integration (SII)*, IEEE, 2015, pp. 211–216.

[133] W.-K. Song and J. Kim, *Novel assistive robot for self-feeding*. INTECH Open Access Publisher, 2012.

[134] A. Yamazaki and R. Masuda, "Various foods handling movement of chopstick-equipped meal assistant robot and there evaluation," *Social Robotics*, pp. 158–167, 2012.

[135] ——, "Autonomous foods handling by chopsticks for meal assistant robot," in *ROBOTIK 2012; 7th German Conference on Robotics*, VDE, 2012, pp. 1–6.

[136] H. Admoni and S. Srinivasa, "Eye gaze reveals intentions in shared autonomy," in *In Proceedings of Intentions in HRI Workshop at HRI*, ACM/IEEE, 2017.

[137] Y. Kobayashi, Y. Ohshima, T. Kaneko, and A. Yamashita, "Meal support system with spoon using laser range finder and manipulator," *International Journal of Robotics and Automation*, vol. 31, no. 3, 2016.

[138] Y. Takahashi and S. Suzukawa, "Easy human interface for severely handicapped persons and application for eating assist robot," in *Mechatronics, IEEE International Conference on*, 2006, pp. 225–229.

[139] G. Canal, G. Alenyà, and C. Torras, "Personalization framework for adaptive robotic feeding assistance," in *International Conference on Social Robotics*, Springer, 2016, pp. 22–31.

[140] C. J. Perera, I. Naotunna, C. Sadaruwan, R. A.R. C. Gopura, and T. D. Lalitharatne, "SSVEP based BMI for a meal assistance robot," in *Systems, Man, and Cybernetics (SMC), IEEE International Conference on*, IEEE, 2016, pp. 002 295–002 300.

[141] C. J. Perera, T. D. Lalitharatne, and K. Kiguchi, "Eeg-controlled meal assistance robot with camera-based automatic mouth position tracking and mouth open detection," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 1760–1765.

[142] D. Park, H. Kim, and C. C. Kemp, "Multimodal anomaly detection for assistive robots," *Autonomous Robots*, submitted.

[143] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 1096–1103.

[144] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989, pp. 257–286.

[145] T. P. Minka, "From hidden markov models to linear dynamical systems," Tech. Rep. 531, Vision and Modeling Group of Media Lab, MIT, Tech. Rep., 1999.

[146] H. Hoffmann, "Kernel pca for novelty detection," *Pattern Recognition*, vol. 40, no. 3, pp. 863–874, 2007.

[147] D. Yu and M. L. Seltzer, "Improved bottleneck features using pretrained deep neural networks.," in *INTERSPEECH*, vol. 237, 2011, p. 240.

[148] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, IEEE, 2013, pp. 3377–3381.

[149] A. Schliep, W. Rungsarityotin, and B. Georgi, *General hidden markov model library*, `http://www.ghmm.org`, 2004.

[150] H. Ketabdar, J. Vepa, S. Bengio, and H. Bourlard, "Using more informative posterior probabilities for speech recognition," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, IEEE, vol. 1, 2006, pp. I–I.

[151] C. M. Bishop, *Pattern recognition and machine learning (information science and statistics)*. Springer-Verlag New York, Inc., 2006, ISBN: 0387310738.

[152] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, p. 1257, 2006.

[153] C. E. Rasmussen, M. Kuss, *et al.*, "Gaussian processes in reinforcement learning.," in *NIPS*, vol. 4, 2003, p. 1.

[154] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*, Springer, 2004, pp. 63–71.

[155] N. A. C. Cressie, *Statistics for spatial data*, ser. Wiley series in probability and mathematical statistics. New York, Chichester, Toronto: J. Wiley & Sons, 1993, ISBN: 978-0-471-00255-0.

[156] A. Jain, M. D. Killpack, A. Edsinger, and C. C. Kemp, "Reaching in clutter with whole-arm tactile sensing," *The International Journal of Robotics Research*, vol. 32, no. 4, pp. 458–482, Apr. 2013.

[157] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, IEEE, vol. 2, 2005, pp. 590–596.

[158] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard, "Yaafe, an easy to use and efficient audio feature extraction software," in *ISMIR*, 2010, pp. 441–446.

[159] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[160] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1867–1874.

[161] Y.-X. Hung, C.-Y. Chiang, S. J. Hsu, and C.-T. Chan, "Abnormality detection for improving elders daily life independent," in *International Conference on Smart Homes and Health Telematics*, Springer, 2010, pp. 186–194.

[162] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic press, 2013.

[163] D. Park, H. Kim, Y. Hoshi, Z. Erickson, A. Kapusta, and C. C. Kemp, "Multimodal execution monitoring for robot-assisted feeding," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, 2017.

[164] D. Park, Y. Hoshi, and C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. PP, no. 99, pp. 1–1, 2018.

[165] D. J. Im, S. Ahn, R. Memisevic, Y. Bengio, *et al.*, "Denoising criterion for variational auto-encoding framework.," in *AAAI*, 2017, pp. 2059–2065.

[166] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

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

[168] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," Technical Report, Tech. Rep., 2015.

[169] M. Madry, L. Bo, D. Kragic, and D. Fox, "St-hmp: Unsupervised spatio-temporal feature learning for tactile data," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014.

[170] P. Ahrendt, "The multivariate gaussian probability distribution," Tech. Rep., 2005.

[171] G. F. Kuhn, "Model for the interaural time differences in the azimuthal plane," *The Journal of the Acoustical Society of America*, vol. 62, no. 1, pp. 157–167, 1977.

[172] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[173] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[174] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[175] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[176] T. Bhattacharjee, A. Jain, S. Vaish, M. D. Killpack, and C. C. Kemp, "Tactile sensing over articulated joints with stretchable sensors," in *World Haptics Conference (WHC), 2013*, IEEE, 2013, pp. 103–108.

[177] P. Rashidi and A. Mihailidis, "A survey on ambient-assisted living tools for older adults," *IEEE journal of biomedical and health informatics*, vol. 17, no. 3, pp. 579–590, 2013.

[178] N. Tejima, "Rehabilitation manipulator for eating," *Journal of the Robotics Society of Japan*, vol. 14, no. 5, pp. 624–627, 1996.

[179] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: Ros for non-ros users," in *Robotics Research*, Springer, 2017, pp. 493–504.

[180] Assistive Innovations, *Ieat feeding robot*, `https://www.assistive-innovations.com` [Accessed: 2017-12-06], 2016.

[181] G. Garcia-Mateos, A. Ruiz, P. E. L. de Teruel, A. L. Rodriguez, and L. Fernandez, "Estimating 3d facial pose in video with just three points," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008, pp. 1–8.

[182] A. Weiss, R. Bernhaupt, M. Lankes, and M. Tscheligi, "The usus evaluation framework for human-robot interaction," in *AISB2009: Proceedings of the symposium on new frontiers in human-robot interaction*, vol. 4, 2009, pp. 11–26.