LEARNING ROBOTIC NAVIGATION AND MANIPULATION PRIMITIVES FROM DEMONSTRATION

by

Yiğit Yıldırım

B.S., Computer Engineering, Boğaziçi University, 2010

M.S., Computer Engineering, Boğaziçi University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2025

ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to my advisor, Emre Uğur, for his immense patience and perpetual support in bringing me to this stage. This journey would not have been possible without his guidance and encouragement.

I am profoundly grateful to İnci M. Baytaş and Barış Akgün for their invaluable feedback and continued support throughout the years of my research. My heartfelt thanks also go to Arzucan Özgür and Hatice Köse for graciously agreeing to serve as jury members in my defense.

I would like to acknowledge my former advisor, H. Levent Akın, who first introduced me to the field of robotics and inspired me to pursue a career as a robotics researcher. His mentorship and belief in me have been pivotal to my growth.

I am also thankful to my colleagues at the Colors Lab, particularly Alper Ahmetoğlu, Ahmet Ercan Tekden, Igor Lirussi, and Yunus Şeker, for their constant support and insightful discussions that have enriched my research.

This thesis was supported by the European Union under the INVERSE project (project number: 101136067) and by TUBITAK ARDEB 1001 program (project number: 120E274). The material created within the scope of this thesis work and whose copyrights are transferred to the publishing house were used in the thesis in accordance with the permission of the publishing house.

Finally, my deepest appreciation goes to my family and my beloved Gizem Ünsal for their enduring love and encouragement. They have been my pillars of strength throughout this long journey. They truly deserve this achievement as much as I do.

ABSTRACT

LEARNING ROBOTIC NAVIGATION AND MANIPULATION PRIMITIVES FROM DEMONSTRATION

The current raison d'être of robots is their potential to be utilized in tasks deemed suitable by humans and for the benefit of humanity. To ensure that robots perform designated tasks as successfully as humans, it is logical to aim for human intelligence or functionality. The research presented in this thesis is a collection of attempts to bring robot functionality one step closer to the human level. Learning from Demonstration (LfD), which essentially signifies teaching robots new skills by demonstrating them, provides invaluable resources for progressing in this direction. With the advances in data-driven approaches, LfD methods have undergone a significant transformation. Despite the improvements, there remains a substantial gap between humans and robots in terms of intelligence. To help bridge this gap, we first proposed increasing the sociability of mobile robots. We introduced a data-driven navigation framework that learns navigation-related movement primitives from real-world human data. The proposed model leverages spatial metrics of proxemics and trajectory characteristics to minimize disturbance to surrounding individuals. Building on this work, we proposed a new LfD framework called Conditional Neural Expert Processes (CNEP). CNEP learns movement primitives of diverse skills simultaneously in an unsupervised manner. Leveraging the entropy concept, CNEP paves the way for capturing the multimodalities in demonstrations. Finally, many real-world skills require the rhythmic application of the same primitive, such as hammering a nail. For such skills, we proposed the Position-Enhanced Movement Primitives (PEMP) model that learns to utilize the angular phase instead of the linear phase. Our experiments showed that PEMP has great potential in modeling and synthesizing periodic primitives.

ÖZET

GÖSTERİM YOLUYLA ROBOTİK NAVİGASYON VE MANİPÜLASYON PRİMİTİFLERİNİN ÖĞRENİMİ

Robotların varlık sebebi, insanların uygun bulduğu görevlerde, insanların yararına kullanılabilecek olmalarıdır. Robotların istenen görevleri, insan kadar başarılı şekilde yapabilmesini sağlamak için insan zekasını veya işlevselliğini hedeflemek mantıklıdır. Bu tezde sunulan araştırmalar, robotların işlevselliğinin insan seviyesine yaklaştırılmasını sağlamak için yapılmış bir dizi çalışmayı kapsamaktadır. Gösterimden Öğrenme (Learning from Demonstration - LfD) tekniği, bu yönde ilerlememiz için paha biçilmez kaynaklar sağlamaktadır. Veri odaklı yaklaşımlardaki son gelişmelerle birlikte, LfD yöntemleri de bir dönüşüm sürecine girmiştir. İlerlemelere rağmen, zeka açısından insanlar ve robotlar arasında önemli bir fark vardır. Bu farkı azaltmaya yardımcı olmak için, ilk olarak mobil robotların sosyal uyumunu artırmayı önerdik. Bu amaçla, navigasyona ilişkin primitifleri, insan verilerinden öğrenen bir navigasyon çerçevesi geliştirdik. Bu model proksemiklere ve yörüngelere ait mekansal metriklerden yararlanarak robot hareketlerinin çevredeki insanlarda yarattığı rahatsızlığı azaltmayı hedefler. Devamında, Koşullu Sinirsel Uzman İşlemler (Conditional Neural Expert Processes - CNEP) adını verdiğimiz yeni bir LfD çerçevesi önerdik. CNEP, farklı hareket primitiflerini aynı anda ve insan müdahalesine ihtiyaç duymadan öğrenir; entropi konseptinden yararlanarak, gösterimlerdeki çok modluluğun yakalanmasının yolunu açar. Son olarak, bir çiviyi çekiçle çakmak gibi birçok gerçek dünya becerisi, aynı primitifin ritmik olarak uygulanmasını gerektirir. Böyle periyodik görevler için, doğrusal faz yerine açısal fazı kullanan Pozisyonla Güçlendirilmiş Hareket Primitifleri (Position-Enhanced Movement Primitives - PEMP) modelini önerdik. Deneylerimiz, PEMP'in periyodik primitifleri modelleme ve sentezleme konusunda büyük bir potansiyele sahip olduğunu göstermiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

b	Batch size in training
C	Configuration of the environment
d	Number of experts in the CNEP model
D	Set of expert demonstrations
d_{goal}	2D vector of the relative position of the current goal
d_{ped}	Array of 2D vectors representing positions of pedestrians
$\mathbb{E}_d$	Expectation on the distribution d
e(t)	Error signal
K	Number of pedestrians
K_d	Differential Coefficient in PID
K_i	Integral Coefficient in PID
K_p	Proportional Coefficient in PID
$\mathcal{L}$	Loss function
$\mathcal{L}_{batch}$	Entropy-based batch loss
$\mathcal{L}_e$	Reconstruction loss for expert e
$\mathcal{L}_{ind}$	Entropy-based mean loss of experts for a trajectory
$\mathcal{L}^i_{rec}$	Cumulative reconstruction loss for trajectory i
m_{max}	Maximum number of targets
n_{max}	Maximum number of observations
$p_{i,e}$	Probability of an expert-trajectory pair
r_{avg}	Average representation in the latent space
T	Maximum time step of a trajectory
t_q	Query time step
u(t)	Control signal
v^x	X component of the velocity vector
v^y	Y component of the velocity vector
α_1	Coefficient of the first loss component

γ	Task (context) parameter
μ_q	Mean prediction at query time
Σ_q	Standard deviation at query time
τ	Trajectory

LIST OF ACRONYMS/ABBREVIATIONS

AA	Average Acceleration
ADE	Average Displacement Error
AGV	Auomated Guided Vehicle
AMR	Autonomous Mobile Robot
ATG	Average Time to Goal
BC	Behavior Cloning
CP	Control Policy
CNP	Conditional Neural Process
CNMP	Conditional Neural Movement Primitive
CNEP	Conditional Neural Expert Process
DL	Deep Learning
DMP	Dynamic Movement Primitive
DNN	Deep Neural Network
DOF	Degrees-of-Freedom
DRL	Deep Reinforcement Learning
FFNN	Feed-Forward Neural Network
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
GP	Gaussian Process
GPT	Generative Pre-trained Transformer
HMM	Hidden Markov Model
IRL	Inverse Reinforcement Learning
IZIC	Intimate Zone Intrusion Counts
LfD	Learning from Demonstration
LSTM	Long-Short Term Memory
ML	Machine Learning
MLP	Multi Layer Perceptron

MoE	Mixture of Experts
MP	Movement Primitive
MSE	Mean-Squared Error
OOD	Out-of-Distribution
PbD	Programming by Demonstration
PDMP	Periodic Dynamic Movement Primitive
PE	Positional Encoding
PEMP	Position-Enhanced Movement Primitive
PID	Proportional-Integral-Derivative
PL	Path Length
ProMP	Probabilistic Movement Primitive
PZIC	Personal Zone Intrusion Counts
RL	Reinforcement Learning
RGB	Red Green Blue
RND	Random Network Distillation
RVO	Reciprocal Velocity Obstacle
SCAND	Socially Compliant Android Navigation Dataset
SFM	Social Force Model
SI	International System of Units
SM	Sensorimotor
W-GAN	Wasserstein Generative Adversarial Network

1. INTRODUCTION

1.1. Core Functionalities for Autonomous Robots

The main goal of programming is to delegate a portion of our responsibilities to artificial agents. Unquestionably, there are many tasks in our daily lives that can be performed by software-only agents, such as finding the best thermostat for our home. On the other hand, a vast majority of the physical world tasks require a physical embodiment, such as installing this thermostat in the home. This is where robots come into the picture. We utilize robots to autonomously perform some of the physical tasks that traditionally humans do.

Autonomous execution in the physical world necessitates providing robots with at least two primitive functionalities: (1) Sensing so that the robot can sense the status of the world and (2) acting so that the robot can take essential actions to perform the required task. Basically, the information produced by the robot's sensors is processed in some way to determine the action the robot would take. The earliest applications, such as [1], employed simple lookup tables that directly mapped the sensory input to corresponding actions. Later, as the technology advanced, roboticists started using more capable control policies (CPs), which are specialized functions mapping environment states, conditions or situations in which the robot operates, to robotic actions.

1.2. Challenges in Real World

It has been apparent since the initial robotic applications that we need better sensors and actuators as well as more proficient CPs to handle the complexity and dynamicity of the real world. Owing to the constant advances in hardware technology, robotic sensors, actuators, and computation units have been continually improving. So do the CPs. Today, advanced computers can process abundant sensory data to pave the way for more dexterous actions than conventional ones. However, it still remains

too ambitious to present our problems to robots and expect them to come up with solutions that are better than ours. The current aspiration of robotics research is to enable them to match the levels of human expertise.

Nowadays, we can safely assert that researchers have a consensus that generalization and scalability are two essential characteristics for CPs to autonomously execute tasks in the real world, as stated in [2]. Data-driven techniques, especially Machine Learning (ML), can help robots acquire these abilities. Rooted in the flocking behavior of ants, Reinforcement Learning (RL) is an ML approach aiming to equip robots with optimal CPs given a specific representation of the problem definition. With the integration of deep-learning (DL) based function approximators, as in [3], RL has achieved revolutionary performances in the last decade. On the other hand, its use in robotics is still largely confined to simulated environments rather than real-world applications. One reason why learning new behaviors in the real world is challenging, as discussed in [4], is that the search space that needs to be explored can be enormously vast. Even so, RL is an invaluable approach, and researchers have been working to provide solutions to eliminate the barriers in front of its widespread application, as in [5].

1.3. Learning from Demonstration (LfD) for Skill Acquisition

Another line of work has adopted an alternative approach to equipping robots with skillful CPs. Instead of describing the goal and letting the robot search for a solution by itself, this line of work focuses on the impracticality of this search in high-dimensional and continuous perception-action spaces and proposes imitation or programming by demonstration [4]. Learning from Demonstration (LfD) and Programming by Demonstration (PbD) are used interchangeably in the literature, and throughout this thesis, LfD will be preferred.

LfD is a teaching paradigm and is limited to neither programming nor robotics. The concept of learning through observation transcends robotics and aligns with cognitive and educational psychology. For example, the concept of Zone of Proximal

Development, introduced in [6], is an analogous concept explaining the role of guidance in improving the cognitive capabilities of the learner. LfD, in broad terms, refers to the procedure that enables the acquisition of new skills by observing an expert. In robotics, LfD signifies the process of acquiring CPs through expert-provided sample demonstrations of the intended policy the robot is expected to learn [7].

1.4. Movement Primitives (MPs)

The complexity of most real-world tasks imposes a significant challenge in creating CPs. Research in neuroscience, such as [8], has illustrated that intricate animal behavior is hierarchical and modular, containing simpler units called motor (or movement) primitives (MPs). The authors state in [9] that complex movements are attained by the combination of these MPs. Analogously, in robotic control, we use the term MPs to refer to the policy units that are modular and can complete individual movement behaviors. They are proposed to facilitate the implementation of complex CPs, as explained in [10]. Therefore, representing the sensorimotor (SM) information in terms of MPs has been considered an important step in generating complex movement behaviors.

Introduced in [11], Dynamic Movement Primitives (DMPs) are one of the prominent methods, providing a mathematical framework to encode expert-demonstrated SM trajectories in terms of MPs. The proposed representation models an entire SM trajectory as a stable nonlinear dynamical system composed of an attractor and a forcing term. Upon modeling, the nonlinear forcing term can be modified to account for perturbations on the modeled trajectory, and the system can synthesize SM trajectories that generalize into novel conditioning points. DMPs are of key importance and have considerably affected the research on LfD techniques. It set the success criteria for many MP-based LfD frameworks proposed afterward, such as [12–16].

One of the important questions that the original DMP formulation does not address is how the modeling of multiple SM trajectories was going to be achieved.

Fortunately, probabilistic and neural-network-based MP techniques offer solutions by modeling distributions of demonstration trajectories. Fortunately, these improvements in the literature provided the necessary frameworks to shape the motivations of this thesis.

1.5. Thesis Motivation and Contributions

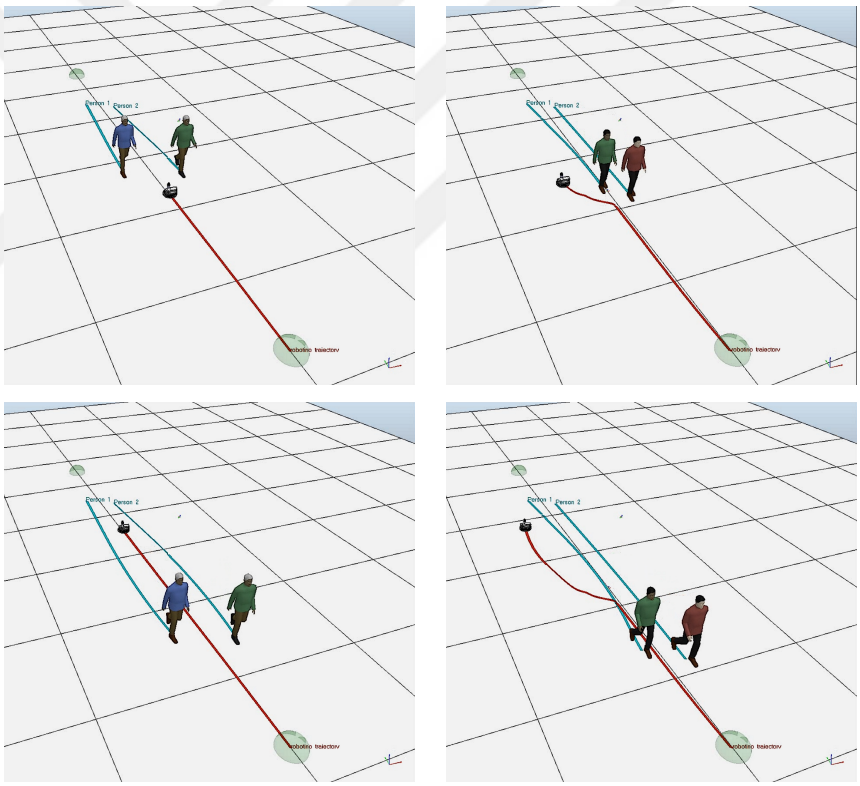
The ultimate motivation for this thesis is to create more skillful CPs for robots to achieve or approximate human-level expertise in both navigation and manipulation tasks. After each study explained below, we proceeded to provide solutions to the problems we experienced during the previous study. However, the reason behind the selection of the first research question was based on the observation of the ever-increasing utilization of Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs) in large production facilities and the possibility that these vehicles could potentially replace human workers. Nevertheless, the shift from human labor to autonomous robots has not proliferated yet. The gradual nature of this change requires humans and robots to coexist in the same environments and collaborate on many tasks. Similarly, a popular aspiration in robotics research is to hand over some household chores to robots. For the people dreaming of having personal robotic assistants, this means sharing their homes or designated environments with robots.

In the next sections, we detail the following contributions:

- We developed a data-driven navigation system to enhance the sociability of mobile robot navigation [17].
- We proposed a novel LfD framework, CNEP, for learning skills from multimodal expert demonstrations [18].
- We proposed the PEMP model, another novel LfD framework for modeling periodic skills.

1.5.1. Social Navigation for Mobile Robots

Any topic in robotics focusing on the social aspects rather than the mechanical or computational aspects can be considered a part of Social Robotics research [19]. Since our physical world is designed considering the needs and capabilities of humans, the robots that share our social environment have to follow some rules. In the case of mobile robot navigation, these rules can be social and cultural norms, such as not approaching another person in an intimidating manner, adjusting its velocity so that the people around feel more comfortable, etc.



(a) (b)

Figure 1.1. Comparison between (a) the regular and (b) social navigation. On the left, the robot passes between a group of two pedestrians, taking an energy-efficient trajectory. This behavior has the disadvantage of disturbing the people encountered on the way. The navigation trajectory on the right prioritizes people's comfort over efficiency. Therefore, it is less disturbing and expected from socially compliant robots.

The concept of social navigation lies in the intersection of two concepts: navigation and human-robot interaction. It describes improving the navigation of the robot to enhance its comprehensibility with humans. Most traditional robot navigation frameworks, such as [20–23], can be considered quite safe. The word "safe" reflects physical safety, such as collision avoidance. On the other hand, Figure 1.1 explains the concept of social navigation. On the left, we see a robot with a perfectly safe but socially non-optimal navigation plan. Although non-optimal from physical aspects, the planned path on the right is socially compliant.

Nowadays, Deep Neural Networks (DNNs) are being used as powerful function approximators across different domains, such as healthcare [24], finance [25], drug discovery [26], etc. Robotics is one of the principal application areas of many developments as it relies on many other areas, such as speech, vision, planning... Additionally, the physical embodiment of robots presents novel challenges, as well as bringing opportunities, paving the way for developing new ML approaches, such as Vision-Language-Action models [27, 28]. Social navigation is one of those areas that is highly benefited by developments in ML. While earlier approaches, such as [29, 30], proposed statistics-based techniques to model the space around humans to decrease the discomfort a robot may impose, more recent approaches, such as [31], come up with spatial affordance maps integrating data-driven techniques further. Modern studies, such as [32–40], highly rely on state-of-the-art ML approaches.

To improve the sociability of mobile robot navigation, we first propose a data-driven navigation architecture to model the navigational behaviors of humans in terms of MPs. The framework uses Conditional Neural Processes (CNPs), introduced in [41], to learn global and local controllers of the mobile robot from observations of human trajectories. Normally, the arbitration mechanism between these two controllers is rather straightforward; the global planner guides the local planner whenever the local planner has no valid targets. On the other hand, one serious weakness of the CNP architecture is that it fails to extrapolate without noticing that it is failing outside the demonstration space. Therefore, we require another module to oversee the operation

of these planners. For this purpose, we leverage a state-of-the-art, deep prediction mechanism called Random Network Distillation (RND) to detect situations not similar to the trained ones and would possibly lead to failure. Upon detecting such states, the robotic base is controlled temporarily by a manually coded module until CNP-based modules are safe again. Our results demonstrate that the proposed framework can successfully perform navigation tasks regarding social norms in the demonstration data. Further, we showed that our system produces fewer personal-zone violations, causing less discomfort. After the initial results are published in [42], the last version of this work with more thorough experiments is published in Interaction Studies [17] and explained in detail in Section 3. Building on our experiences, the social navigation framework is still being improved with the addition of temporal modeling mechanisms. The current status of these efforts is presented in [37].

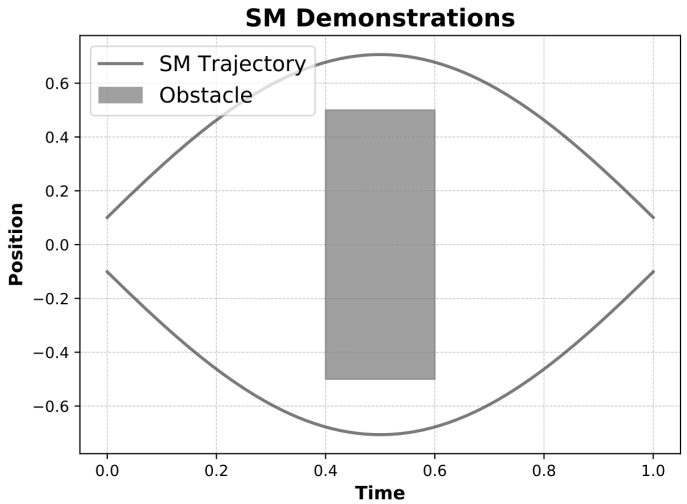


Figure 1.2. Environment configuration for an imaginary obstacle avoidance task and two expert demonstrations. Obstacle avoidance in this configuration has a bimodal solution. Expert preference is the only reason for the selection of one of the modes.

1.5.2. Modeling MPs from Multimodal Demonstrations

One of the main difficulties we experienced while working on [17] was handling trajectories with the same shape but different lengths. Two spatially similar trajecto-

ries might indeed imply different behaviors. If two people follow the same trajectory with different velocities, one might be wandering around while the other is in some kind of rush. This minor difference is very important in terms of the socialness of their behavior. When people are in haste, they tend to ignore others. Similarly, in manipulation tasks, take obstacle avoidance as an example; there may be clearly separated SM trajectories due to differences in demonstrators' preferences. This phenomenon is explained in [43] and illustrated in Figure 1.2. In such cases, moment-matching approaches may fail because of the mode-collapse problem; they aim to model the mean of various modes in the training data. However, the mean of multiple solutions might not be a feasible solution and could result in collisions, as explained in [44]. This scenario is reproduced with CNMP and CNEP models on a real-world obstacle avoidance task, and visualization is provided in Figure 1.3.

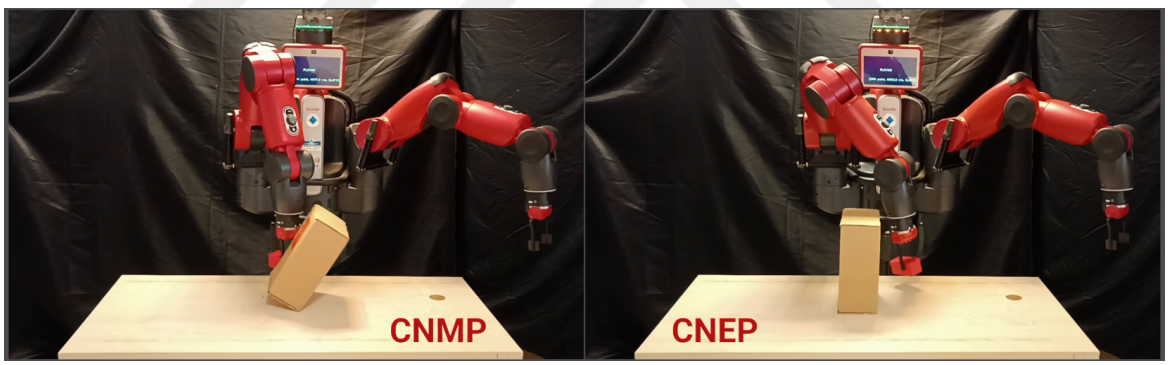


Figure 1.3. Comparison between the CNMP model and the CNEP model in an obstacle avoidance task. Since obstacle avoidance is a problem with a solution of bimodal SM trajectories, CNMP fails to synthesize appropriate answers while CNEP successfully learns this multimodal nature of the expert demonstrations.

To make a distinction between different modes in the given SM trajectories and model each mode separately, we propose a novel MP-modeling framework called Conditional Neural Expert Processes (CNEPs) in [18]. The CNEP model, as explained in Section 4, utilizes multimodal demonstration trajectories by leveraging a mixture-of-experts (MoE) architecture and entropy-based arbitration among them. This work is published in IEEE Robotics and Automation Letters.

1.5.3. Modeling Periodic Skills

There are many periodic skills in the real world that require the rhythmic application of the same control signal over and over. Although the CNEP model offers improved performance compared to many other MP methods across a variety of tasks, we noticed that it required too much effort when the model was asked to encode a periodic movement trajectory. The reason behind this weakness is that CNEP treated every single point on the trajectory equally and did not leverage the rhythmic nature of the behavior. This is a common problem for approaches that assume linearity in the phase of the motion. On the other hand, using the angular phase provides the opportunity to transform the problem into an easier space and streamline the modeling process for cases, as shown in Figure 1.4. The idea is borrowed from Periodic DMPs, introduced in [45], and still being improved in recent studies, such as [46].

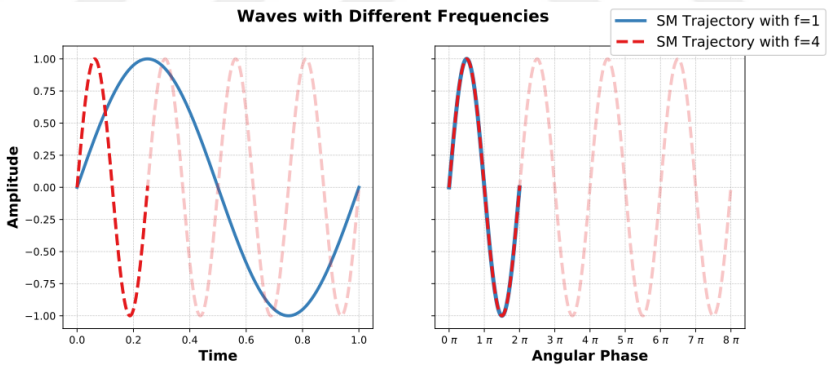


Figure 1.4. An example scenario where the utilization of angular phase provides an opportunity to transform the modeling problem into a simpler space.

The assumption of linearity in the phase of the motion undermines the capabilities of the underlying neural architecture, which we observed to be capable of discovering periodic relations leveraging an oscillatory phase. Therefore, instead of using the absolute positions of the SM values as a linear phase variable, we propose the application of the positional encoding method, which was originally introduced in [47]. We explained the proposed model in Section 5 and submitted the corresponding paper to IEEE Access.

The remainder of the thesis is structured as follows: we begin with a literature review in Section 2. Subsequently, we discuss the relevant studies in their respective sections. Section 6 presents an overall discussion, and we give our final remarks in Section 7.

2. RELATED WORK

In robotics, Learning from Demonstration (LfD) is a paradigm where robots learn to perform tasks by observing expert demonstrations rather than being explicitly programmed. Early work in LfD-based robotic control focused on directly recording and replaying sensorimotor trajectories of target tasks, leading to robust and rigid controllers. Nevertheless, these handcrafted controllers usually fail to adapt to unseen conditions because they rely on exact trajectory replication [48]. Instead, it is evident from comparative studies that using data-driven techniques to allow robots to generate their own policies leads to learning better representations and eventually to more adaptable policies [49]. Therefore, incorporating data-driven approaches to LfD frameworks enables learning generalizable skills without detailed, task-specific programming, which is advantageous as it implicitly captures the expert's knowledge through demonstrations. In the following, we first give a literature overview of the social navigation concept, and then, in movement primitives.

2.1. Social Navigation

Social navigation implies the exhibition of socially competent behaviors during the robot's navigation. The nonverbal interaction caused by the robot's navigation may be improved by the integration of social and cultural norms that people follow. In [50], the authors describe the benefits of social navigation as follows: it increases the comfort of the people around the robot, improves the naturalness of the robotic platform, and enhances the sociability of the robot. The concept of social navigation lies in the intersection of two fields: navigation and human-robot interaction. As presented in the left column of Figure 1.1, a robot with a perfectly safe navigation plan might disregard the importance of the comfort level of the encountered pedestrians. In contrast, although non-optimal, the executed navigation trajectory on the right is more socially compliant since it cares about the comfort level of the people around it.

The early works in the domain are influenced by studies in social sciences. The concept of proxemics, introduced in [51], defines abstract social zones around the people and provides a basis for many studies in socially-compliant robot navigation [52–56]. Although these pioneer studies relied on a predefined set of rules to achieve social navigation, they drew attention to the subject and made it more popular.

Another important study in social sciences is the [57], where researchers introduce the Social Force Model (SFM) to explain the navigational behaviors of humans. They define a set of functions to calculate the local movements of pedestrians to reach a global goal. The simplicity of the SFM model causes many researchers in robotics to adopt the approach of moving the robots as humans do [58–60].

As of late, data-driven approaches have become more prevalent in explaining human behavior since they are more adaptable to many situations. Many researchers, as in [61–63], apply Inverse Reinforcement Learning (IRL) to calculate a reward function that describes the navigational behavior of the human. Another stream of work uses Deep Reinforcement Learning (DRL) to generate human-aware robot navigation, such as [64]. These studies assume the availability of either the reward or the features that compose the reward. To relax this assumption, approaches that learn the social norms merely from the data became more popular.

In [65], researchers use networks with Long-Short Term Memory cells (LSTMs). Later, an improved version of this study is presented in [66], where researchers use LSTMs inside a generative adversarial setting. Similarly, in [67], an encoder-decoder architecture relying on internal LSTM modules is used. These studies successfully predict human navigation trajectory in a limited local frame. Despite being considerably close to the social navigation domain, the trajectory prediction methods in these studies cannot be directly applied to mobile robots. A robot placed in a real-world environment has to meet serious time complexity considerations. Moreover, these approaches are limited to local frames, while path-planning on mobile robots requires a global navigation goal, as explained in [68].

In an environment, the density of the human population may substantially affect navigational behaviors. Researchers build a system of simple hand-crafted utility functions to address this factor to generate social navigation in [69]. This approach successfully generates social navigation in highly crowded environments. In addition, using simple utility functions is a fruitful technique for handling many surrounding pedestrians, which may be difficult for data-driven strategies due to computational limitations. However, this approach is prone to the same limitations as other hand-crafted techniques.

On the contrary, many other studies, such as [70] and [71], intend to capture the underlying behavior of individuals instead of using a holistic approach. They offer grouping or leader-following behaviors in highly crowded scenarios. This aligns with the taxonomy of navigational behaviors presented by [67], which categorizes the types of navigation interactions into four classes: grouping, leader-following, collision-avoidance, and others. We believe that targeting distinct classes of interactions separately is a better alternative because agents take different actions in different types of social encounters. For example, maneuvering with a group has different dynamics than maneuvering individually to avoid a probable collision. While our study explicitly targets interactions requiring individual collision-avoidance behavior, it can also be modified or extended with representational changes to handle different types of interactions.

Data-driven approaches generally process navigation trajectories in datasets to realize social navigation. However, as mentioned in [72], the scarcity of established datasets has been a significant issue in the domain. Until recently, many studies in the literature (e.g. [73,74]) have been using pedestrian datasets to train models, such as [75] and [76]. Although such datasets provide real-world interactions among pedestrians, they do not contain human-robot interaction. Therefore, pedestrian datasets may overlook any possible effect of an embodied robotic agent's presence on a neighboring pedestrian's navigation trajectory. Lately, we have witnessed an increase in datasets focusing on human-robot interactions for social navigation studies. In [77], researchers

aim to evaluate the comfort levels of pedestrians around a robot in an environment. On the other hand, this dataset does not capture the navigation dynamics of the robot and pedestrians, as it merely focuses on static scenes. Recent datasets, such as [78–80], address both shortcomings, where researchers control robots by teleoperation in real-world environments to record navigation demonstrations.

2.2. Movement Primitives (MPs)

Equipping robots with the desired skills has been the driving force in robotics research. In initial studies, controllers with precise mathematical representations were used. These representations were formed using the physics-based dynamic models of the environment and the kinematic models of the agents [81]. Although accurate in controlled settings and computationally less intense, the applicability of the precise models was limited in realistic scenarios. This is mainly due to their constrained flexibility in the kinodynamic space of the system, preventing the generalization of acquired skills into novel conditions.

MP formalism offers a compact and modular representation to create more flexible controllers. In the LfD setting, one of the most popular MP frameworks is the Dynamic Movement Primitives (DMP). In DMPs [10], expert demonstration of a complex skill is encoded with a system of differential equations in the form of MPs. DMPs can be queried upon modeling to generate motion trajectories from start to end. Also, when integrated with closed-loop feedback, DMPs are proven suitable for real-time control as they adapt to changes and perturbations in real-time, offering robust performance in many applications [82]. However, only a single trajectory can be encoded by the classical DMP formulation, indicating that variabilities inside demonstrations are not considered.

Probabilistic approaches have been proposed to address the abovementioned requirements by offering flexible and robust modeling mechanisms. In this respect, Gaussian Mixture Models together with Gaussian Mixture Regression (GMM-GMR) and

Hidden Markov Models (HMM) have been used in several studies [83–85] to capture the variability of the task by learning the distributions of the demonstration data. The complexity of training and inference in HMMs increases as the dimensionality of the demonstrations increases, whereas variants of GMMs work well with high-dimensional data [86]. Nonetheless, when the demonstration data of the task is sampled from a multimodal distribution, GMMs fail to select one of the modes. In contrast, state-transition probabilities of HMMs encode this information, enabling the synthesis of expert-like trajectories [87].

As stated in [88, 89], the uncertainty in real-world tasks has been explicitly addressed using Gaussian Processes (GPs). As a result, the computational efficiency of learning adaptable and robust robotic controllers is improved to enable control in real-world tasks. Pure GP approaches are known to work well in Euclidean spaces. However, when the demonstration data displays non-Euclidean characteristics, such as rotation of robotic joints, further adjustments are required for GP methods [90].

Addressing the in-task variability, Probabilistic Movement Primitives (ProMP) have been proposed to encode a distribution of trajectories [12]. In [91], ProMPs were shown to provide improved generalization capabilities, enabling generated trajectories to be adapted so that they could pass through desired via points. However, ProMPs are composed of linear Gaussian models limited to unimodal distributions and cannot represent multimodal datasets. Additionally, ProMPs cannot be efficiently trained or queried with high dimensional input.

Using neural networks to encode MPs has led to significant advancements in LfD. Neural networks can learn complex, nonlinear mappings from high-dimensional inputs to motion trajectories, making them well-suited for modeling intricate tasks. In [92], researchers couple DMP with Convolutional Neural Networks to enable DMP learning from multiple demonstrations. Conditional Neural Movement Primitives (CNMPs), which are developed based on Conditional Neural Processes (CNPs) introduced in [41], show great promise in generating flexible and adaptive motion patterns from a limited

set of demonstrations, as they can reconstruct motion trajectories conditioned on a set of via points [16]. The authors of [93] suggest using Contrastive Learning to emphasize stability in the generation of trajectories. In [94] and [18], researchers use entropy to avoid mode collapse in case of multimodal expert demonstrations. These approaches do not address the periodic tasks. Contrarily, the PEMP model manages to model periodic SM trajectories leveraging the oscillatory phase value and nonlinearity induced by the neural network.

Recently, the Stable Movement Primitives (Stable MP) [93], is proposed to learn movement primitives, potentially belonging to multiple skills, using the same neural mechanism. It offers the advantage of guaranteed precision at conditioning points. However, it requires supervision about the type of skill it learns or generates. Additionally, the system limits the conditioning mechanism.

3. LEARNING SOCIAL NAVIGATION FROM DEMONSTRATIONS WITH CONDITIONAL NEURAL PROCESSES

3.1. Introduction

Researchers have been studying mobile robot navigation for decades. Many notable techniques have been proposed in this area over the years, such as [95–97], where safety and robustness features have been prioritized. In other words, the principal driving factor behind the development in this field has been collision avoidance [20]. On the other hand, as humans start to share their environment with robots, new requirements for mobile robot navigation have emerged.

Physical and mental aspects of safety were separately evaluated in [98]. This separation reveals the need to question the psychological efficiency of the navigation systems of mobile robots. To ensure the smooth integration of robots into human environments, these systems must be social and as natural and understandable to humans as they are safe.

The authors of [50] define social navigation as navigating the robot in such a way that minimizes the annoyance that its motion produces. Efforts to decrease the anxiety of pedestrians interacting with a navigating robot can be included in the social navigation domain. To this end, the majority of the studies in the literature target to increase the comfortableness of the interaction. In [99], researchers assert that people find it more comforting to interact with machines same the same way they interact with other people. Therefore, many researchers have aimed to decrease the discomfort that robot navigation generates by replicating human navigation with mobile robots.

The studies in the literature that aim to imitate human behavior in mobile robot navigation fall into two categories: manually coded controllers and learning-based ones.

Manually coded controllers rely on hand-crafted optimization functions to resemble the motion of robots to that of humans. One of the notable studies of this category is the Social Force Model (SFM) [57]. Based on behavioral techniques from the social sciences, SFM suggests that pedestrians move under the effect of specific abstract forces, just like particles in an electric field. While the navigational goal attracts the pedestrian, obstacles and other people exert repulsive forces. Despite its wide application [58–60], some researchers argue that hand-crafted models have limited applicability in controlled environments [62] and that they are not general and applicable to different, varying social environments, especially during avoidance maneuvers [100]. In real-world scenarios, the social compliance of robot navigation requires adaptability. The authors of [101] proposed the use of data-driven approaches to create such adaptive controllers. Researchers have used numerous machine learning algorithms to create better adaptive, socially compliant navigation frameworks. One of the most popular algorithms is Inverse Reinforcement Learning (IRL) [61–63, 101]. Given perfect expert demonstrations, IRL attempts to identify the underlying reward structure, which can be used by any Reinforcement Learning (RL) algorithm to create a human-aware navigation policy. The advantage of this approach is that the reward function is not manually determined but is a linear combination of a set of predefined features. However, the linearity assumption is considered a strong assumption in [102].

Nonlinear rewards can better describe complex behaviors in many real-world problems [103]. Researchers have been using deep learning techniques to leverage this potential in social navigation. In [64], Deep Reinforcement Learning was used to obtain a socially plausible navigation policy. As with other RL approaches, this procedure relies on a predefined reward. Similarly, [102] extracted nonlinear rewards, assuming that the features shaping the reward function are known. Even though this is a relaxation to the known reward constraint, it is very difficult to determine the components of abstract concepts such as socialness. On the other hand, Imitation Learning attempts to learn policies directly from the data, relaxing assumptions about the reward or its features. In [104] and [66], Generative Adversarial Networks were used for direct policy learning from demonstrations. These approaches provided advanced solutions to over-

come the limitations mentioned above. However, these models required too much data for training [105]. On the other hand, learning from a small data set and generalizing to new configurations are desirable.

We also observe that most of the studies on the social navigation domain target only designing/learning the local controllers of the robots since they are responsible for producing motion commands. However, using only the local controller makes the robot vulnerable to the local minima problems [106], and might fail to navigate the robot to its target position. Today, typical robotic navigation systems adopt the two-layered hierarchical approach for path planning tasks [107]. A robot first calculates a trajectory in the so-called global planning phase given an environment. Then, the robot follows this trajectory with a controller in the so-called local planning phase.

This study proposes a novel approach built on top of state-of-the-art neural network architecture, namely Conditional Neural Processes (CNPs) [41]. Given multiple demonstrations of a task, CNPs can encode complex multi-modal trajectories. CNPs extract prior knowledge directly from training data by sampling observations and predicting a conditional distribution over any other target points. Taking advantage of these capabilities, we extended CNPs in two dimensions: to generate complete navigation trajectories in the global planning phase and to generate goal-directed behaviors while actively avoiding pedestrians in the local planning phase. At both levels, our approaches produce trajectories that show the characteristics of the demonstrated ones. They can learn complex, nonlinear, and temporal relationships associated with external parameters and goals. Like other neural network-based learning systems, our system may fail to generate trajectories or control signals when it faces very different situations from the experienced ones, i.e., when it is required to extrapolate to novel situations outside the training range. To detect and react to conditions that may lead to failure, we propose continuously monitoring the environment using a failure prediction system, detecting situations outside the training range and falling back to a hand-crafted reactive controller in case extrapolation is detected. We verified our system in a simulated mobile robot in different environments with static and moving pedestrians.

3.2. Methodology

This work proposes a hybrid data-driven navigation system that uses advanced neural techniques in global and local layers coupled with a novel execution monitoring module for fault-tolerant intelligent navigation. Figure 3.1 demonstrates the overview of our proposed system. Our navigation system consists of four modules: Data-Driven Global Controller, Data-Driven Local Controller, Failure Prediction, and Hand-Crafted Reactive Controller. After introducing these modules in this section, the following sections will provide the details.

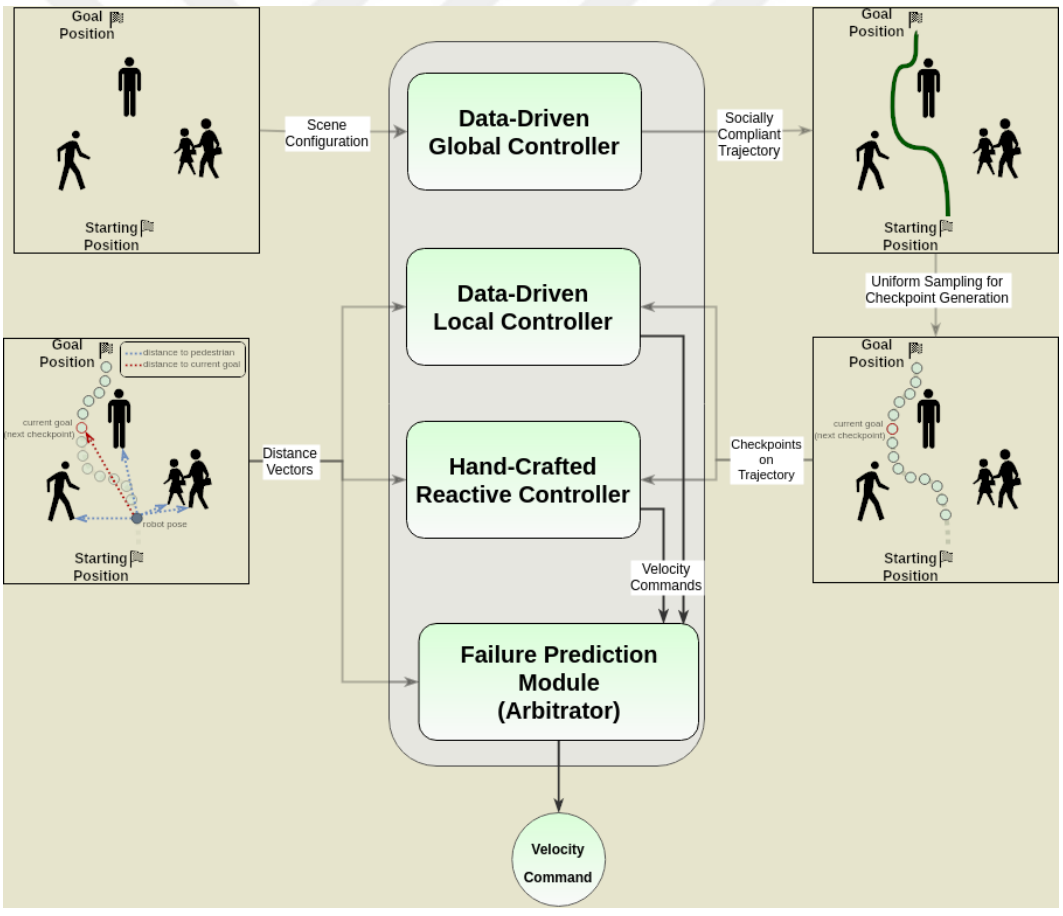


Figure 3.1. The Data-Driven Navigation System is composed of four modules: Data-Driven Global Controller, Data-Driven Local Controller, Failure Prediction, and Hand-Crafted Reactive Controller. Refer to the text for details on the procedure.

First of all, given a target position, the Data-Driven Global Controller is responsible for producing a complete navigation trajectory consistent with the demonstration trajectories previously provided to the robot. Next, several via-points are uniformly sampled from the generated navigation trajectory. The second module, namely the Data-Driven Local Controller, is responsible for generating motion commands to reach each via-point one by one. While maneuvering the robot, this controller reacts to the dynamic changes in the vicinity of the robot and changes the target to the next via point as soon as the current target is reached. This procedure continues until the robot reaches the final target, i.e., the given goal position. Suppose the local controller fails to follow the trajectory at any point; a pedestrian may be blocking the path, for example. In that case, the global controller can be called again to recalculate a new trajectory on the same navigation task. The first two modules use data-driven approaches, which learn from provided trajectories and can be used to learn social competencies for social navigation. The proposed navigation system achieves learning and control in these modules using a neural network family. Similar to other neural network architectures, while models can interpolate to novel situations within the training range, the learned models are prone to extrapolation errors when they run on inputs outside their training range. If unattended, such errors can cause collisions with pedestrians during navigation. Two additional modules are incorporated into our system to detect and react to the extrapolation cases before any collision or failure happens: the Failure Prediction Module and the Hand-Crafted Reactive Controller. The Failure Prediction Module observes the entire operation of the system and is responsible for detecting outlier situations. If this module detects that data-driven controllers are queried with an input outside their training range, the control is temporarily transferred to the Hand-Crafted Reactive Controller, ensuring the safety of navigation. When the risk of failure vanishes, i.e., the robot and its environment are detected to be in the training range again, the data-driven modules take back the control of the robot. These components are combined to form a full-fledged navigation pipeline. In the following, we explain these modules in detail.

3.2.1. I - Data-Driven Global Controller

In our work, the task of global planning is to generate a complete trajectory for navigation. A set of discrete via-points can be sampled from this trajectory to enable the local planner to reach each via-point successively. The Data-Driven Global Controller aims to learn a parametric distribution of social navigation trajectories that reflect the social norms of the people. Therefore, after learning the underlying distribution of socially acceptable trajectories, this module can reactively change the navigation trajectory according to the configuration of the environment.

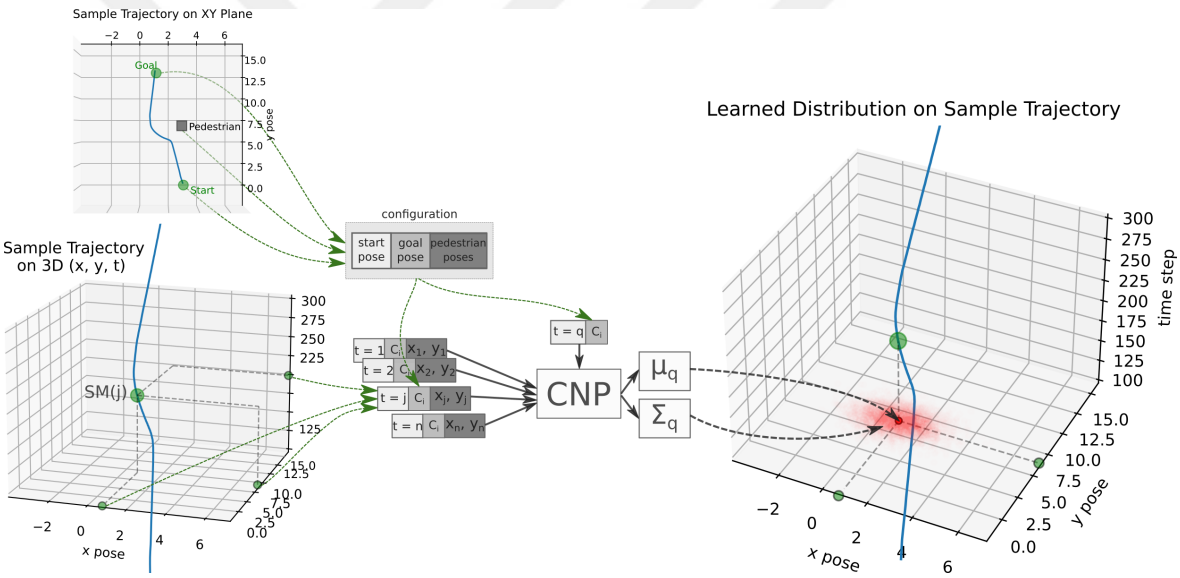


Figure 3.2. Training the Data-Driven Global Controller with a demonstration trajectory. The configuration of the environment and the demonstration trajectory in this environment are gathered and processed. Later, this data is given as input to the CNP structure. The model predicts a bivariate normal distribution of 2D positions for a queried time step. Refer to the text for details.

3.2.1.1. Training the Data-Driven Global Controller. First, we recorded a set of expert demonstrations to teach the trajectory generation function to our model. Each demonstration contains an environment configuration and a navigation trajectory close enough to the optimal social behavior in the corresponding environment. The environ-

ment configuration is the concatenation of the following three elements: the robot's start position, the given goal position, and the pedestrians' positions. In addition, navigation trajectories are stored as a set of (x, y) coordinates with corresponding time steps.

Formally, the set of all trajectories is a collection of N expert demonstrations, denoted by D = $\{\tau^i\}_{i=1}^{N}$. Each navigation trajectory τ is a set of 2D positions at each time step; $\tau = \{(x_j, y_j)\}_{j=1}^{T}$. Furthermore, the configuration C of each scene is formed as C = $\{(x_{start}, y_{start}), (x_{goal}, y_{goal}), ((x_{p_1}, y_{p_1}) \dots (x_{p_K}, y_{p_K}))\}$, where (x_{p_k}, y_{p_k}) represents the 2D position of one of the K pedestrians in the scene.

The overall training procedure is depicted in Figure 3.2. At training time, the system uniformly samples a navigation trajectory τ^i from D. Afterward, n observation points are uniformly sampled on this trajectory, where n is a random number between 0 and n_{max}. The navigation trajectory is the sensorimotor function that the model learns in our representation. It is a function of time and is shaped by the scene's configuration. The realizations of this function on different time steps correspond to 2D positions in the real world. That is, $\tau = \{\text{SM(t)}\}_{t=1}^{T}$, where SM(t) = ($x_t$, y_t). Moreover, we represent the configuration, C, of the environment with the task parameter using the model's γ function. Hence, in this representation, γ(t) = C.

Figure 3.3 introduces the underlying neural network model with the corresponding Encoder-Decoder structure. The Encoder Network takes in (t, γ(t) and SM(t)) tuples, produces their corresponding latent representations, and applies an averaging operation to generate a general representation to encode the n tuples. On the other hand, the Query Network is responsible for generating the distribution related to the position of the robot (SM(t_q)) at any time point t where the generated position is required to be consistent with the average latent representation of n tuples. Therefore, given the average latent representation of n (t, γ(t) and SM(t)) tuples and a random target time point t_q, the Query Network produces the distribution of the predicted position of the robot, i.e., outputs a bivariate normal distribution with parameters (μ_q, Σ_q). With this

output, the loss in the prediction of the complete neural network model is computed as

$$\mathcal{L} = -\log P(\, SM(t_q) \mid \mu_q,\ softmax(\Sigma_q)\,), \tag{3.1}$$

where SM(t_q) corresponds to the actual (x, y) coordinates of the trajectory at time t_q.

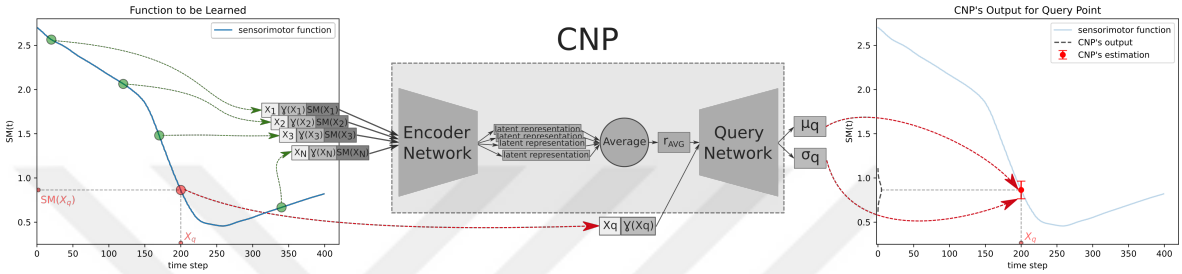


Figure 3.3. The general layout of the training phase of our model. First, a random number of observation points are sampled randomly on SM trajectory. Observation points are fed to the Encoder Network. The query network is run with a random query point to produce the prediction of the system. This prediction causes a loss that is back-propagated through both networks.

3.2.1.2. Querying the Trained Data-Driven Global Controller. After training the Data Driven Global Controller with given navigation trajectories, it can be queried to generate new navigation trajectories given new environment configurations, as illustrated in Figure 3.4. The underlying system can predict the position of the robot at any time point. Therefore, the time step is used as the input, while the scene configuration is used as the task parameter to generate the corresponding position of the desired trajectory. The Data-Driven Global Controller can be queried simultaneously and independently for all time points. In the end, this module outputs an entire trajectory of 2D coordinates - from the starting position to the goal position- as a function of time. When trained with social navigation trajectories, the learned function is a social trajectory generator that avoids pedestrians in the intended course of navigation.

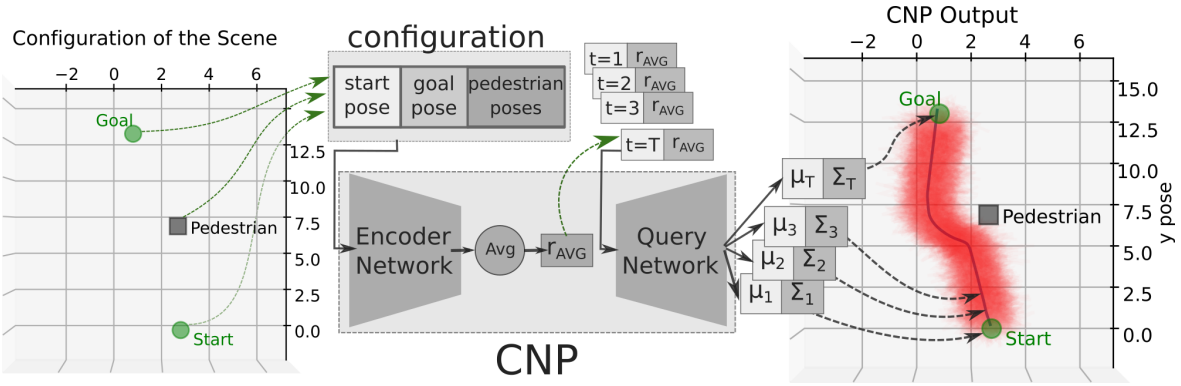


Figure 3.4. Generating an entire navigation trajectory for the configuration given on the left panel. This configuration is given as input to the network, as shown in the middle column. The predicted trajectory is shown on the right. Note that the solid line corresponds to the sequence connecting the mean positions at each queried time step, and the red-shaded area illustrates the variance information.

3.2.2. II - Data-Driven Local Controller

Typically, the main task of a local planner is to move the robot through the via points on the navigation trajectory computed by the global planner. Moreover, it is the task of the local planner to navigate the robot without colliding with pedestrians. In our system, the Data-Driven Local Controller is used to accomplish both tasks and preserve the characteristics present in the demonstrations.

This module is built on top of a CNP-based neural architecture. As in traditional local planners, this module generates goal-directed behavior by targeting short-distance via points as intermediate goals. It uses the relative position of the next checkpoint as input, X. To achieve collision-free navigation, the relative positions of pedestrians are passed to the network as a task parameter, γ(X). This gives the Data-Driven Local Controller the ability to reactively adjust its output with respect to the changing pedestrian positions.

In addition to the basic tasks, the neural network within the Data-Driven Local Controller can learn the social characteristics present in the local evasive maneuvers of

the people, using only the data from their navigation trajectories. In the end, the Data-Driven Local Controller decides which action to take in the form of velocity commands of the mobile base.

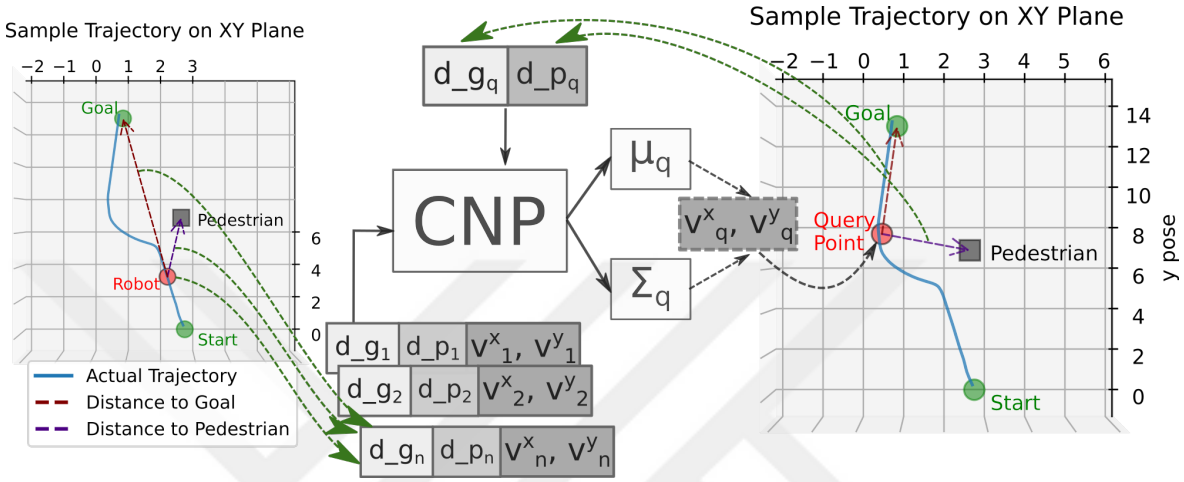


Figure 3.5. Training the Data-Driven Local Controller with a demonstration trajectory. Each observation contains the relative position of the goal, d_g relative positions of the pedestrians, d_p (or d_{ped} throughout the text), and the velocity command, v. The velocity prediction of the model for the query point provides the loss, which is backpropagated through both networks of the CNP structure.

3.2.2.1. Training the Data-Driven Local Controller. The local planners use sensory information to understand their local frames. Our system processes the low-level sensory information to obtain high-level parameters, such as the relative position of pedestrians and the relative position of the goal position. The Data-Driven Local Controller reactively responds to the changes in these high-level parameters. More importantly, it does so while preserving the characteristics present in the previously learned demonstrations. Therefore, when the module is trained with socially acceptable trajectories, it can learn social norms. The demonstration trajectories are recorded as a set of tuples (d_{goal}, [d_{ped}], v), where d_{goal} is a 2D vector of the relative position of the current goal, [d_{ped}] is an array of 2D vectors representing relative position of the pedestrians in the scene, and v is a 2D vector representing the velocity of the base on the 2D plane.

Prior to the training, trajectories are processed and converted into a set of observations that the neural network of the Data-Driven Local Controller uses. An observation is formed by the concatenation of three parts: X, γ(X) and SM(X, γ(X)), where X = d_{goal}, γ(X) = [d_{ped}], and SM(X, γ(X)) = (v^x, v^y), and v^x, v^y correspond to X and Y components of the velocity vector. Figure 3.5 illustrates the training procedure. The training process uniformly samples a trajectory τ^i from the set of demonstrations, D. n observations are uniformly sampled, where n is a random number between 0 and n_{max}. These observations are fed into the Encoder Network to obtain latent representations, which, in turn, are averaged to create the average latent representation of the entire trajectory. This is given as input to the Query Network, along with a random query point X_q, and the value of the task parameter for this query point, $\gamma(X_q)$. The Query Network outputs a bivariate normal distribution with parameters (μ_q, Σ_q), which represents the normal distribution that describes the model's prediction about the velocity command. The loss calculation is the same as in Equation(3.1), which is backpropagated through the entire CNP structure.

Note that in the previously described Data-Driven Global Controller, latent representation encoded the entire trajectory for the corresponding environment and the navigation task and was conditioned with successive time points (independently) to generate the successive points of the trajectory that the robot is supposed to follow. In the Data-Driven Local Controller, on the other hand, the latent representation encodes instantaneous control commands rather than the entire trajectory. It, therefore, is conditioned on the relative position of the checkpoint to generate the corresponding velocity command.

3.2.2.2. Querying the Trained Data-Driven Local Controller. After training encoder and query networks, the model can be run for specific scene configurations. Conditioned on the start and goal positions, the use of the task parameter γ(X) gives the model the ability to reactively change the velocity commands with respect to changing pedestrian positions and possibly changing checkpoint positions in case of the re-computation of the trajectory. At any time, the relative position of the robot with

respect to the current goal (d_{goal}) and the closest pedestrian (d_{ped}) can be concatenated and given to the model as input. The neural network outputs the velocity command (v^x, v^y) to be executed by the robot. At test time, the query is fast enough to be used in real-time applications. Therefore, this module can be used as a local controller that generates velocity commands following the characteristics of the demonstrations, which is used for social navigation.

3.2.3. III - Failure Prediction Module

The previously described Data-Driven Global and Local Controllers learn representations inside the domain they are trained. Therefore, these modules are inherently prone to extrapolation cases when queried outside the learning range. These cases may lead to anomalous behavior and, in practice, generate undesirable velocity commands. Although infrequent, such anomalies may cause collisions and are unacceptable in navigation systems where the safety of pedestrians is of the utmost importance.

To avoid such undesired behavior, we provide the navigation system with the capability to continuously observe the states it encounters during a navigation task for their likelihood of lying outside its training domain. Firstly, we leverage the ability of Generative Adversarial Networks (GAN) [108] to learn the input-space distribution. There are many variants of GANs in the literature, such as Wasserstein GAN (W-GAN) [109]. In our case, standard GAN worked more stably compared to the W-GAN.

Figure 3.6 depicts the training procedure. On the same dataset that the local CNP is trained, we train a GAN. GAN takes as input [d_{ped}] from actual trajectories from the dataset. The generator network attempts to produce realistic candidates to deceive the discriminator network in deciding whether its input belongs to the actual dataset or not. The objective function follows the standard minimax loss introduced in [108]

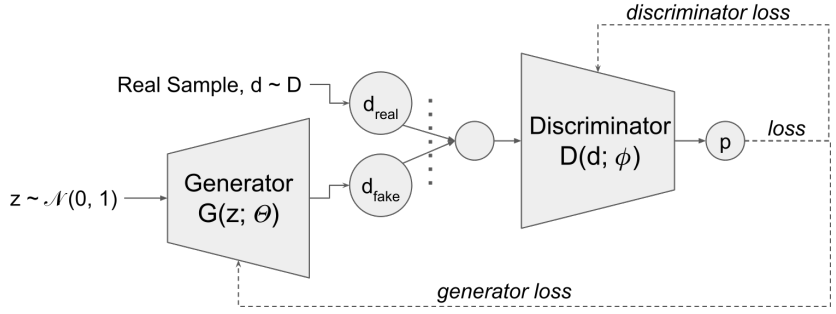
$$\min_G \max_D \mathbb{E}_d \left[\log D(d)\right] + \mathbb{E}_z \left[\log \left(1 - D(G(z))\right)\right]. \tag{3.2}$$

During the training of the networks with their corresponding losses, given in Equation(3.2), the generator network becomes better at producing realistic candidates, and the discriminator network becomes better at discriminating against them. After the training, the discriminator learns the distribution that the actual data belongs to. We use the output of the discriminator network, which is the probability of an encountered state being sampled from the same distribution as the training data, in predicting the extrapolation.

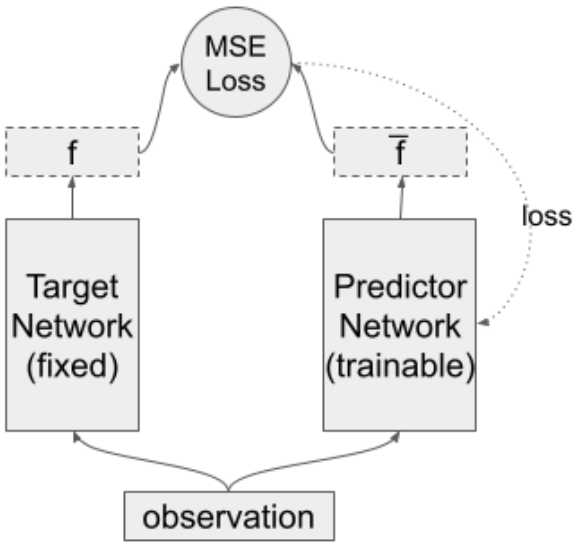
Although the system benefits from using GANs, these architectures are often criticized for being unstable due to oscillation and mode collapse [110]. Therefore, we have implemented another approach, named Random Network Distillation (RND) [111] in this module for detecting out-of-distribution (OOD) samples. This approach uses two sub-networks; the first one - the target network - is a fixed and randomly initialized multi-layer perceptron (MLP), and the second - the predictor network - is a standard fully connected MLP.

The approach evaluates the novelty of an encountered system by calculating the distance of the output that two sub-networks produce. Initially, the two would produce different results. By training the predictor, it starts to output similar values as the target network does. Upon training, when an OOD input is queried, two sub-networks output very different values, producing a high error value and enabling our system to predict extrapolation errors before they occur. The structure of this architecture is given in Figure 3.6.

The model is trained with the relative positions of pedestrians and the goal at each time step. This information is extracted from the same expert demonstrations that the data-driven controller is trained with. After training the RND model, the distance between its sub-networks is used to detect whether a new data point is from the training range.



(a)



(b)

Figure 3.6. Approaches used in the Failure Prediction Module. As shown in (a), GAN learns to attribute small probabilities to cases with unusual pedestrian positions. Contrarily, in (b), the target network outputs random values, and the predictor network learns to mimic them. Instead of GAN, the RND approach is preferred in this study due to its stability.

3.2.4. IV - Hand-Crafted Reactive Controller

Whenever the Failure Prediction Module outputs a small probability for an observation point, the navigation system concludes that it is about to extrapolate. This usually leads to erratic movements and potentially a collision with a pedestrian. To control the robot safely in those situations, we created the Hand-Crafted Reactive Controller. This module implements the Social Force Model (SFM) to move the robot in a

safe and socially compliant manner. On the other hand, the SFM is not flexible enough as many real-world applications require since the attractive and repulsive components are defined and tuned manually. Therefore, the pipeline uses the Hand-Crafted Reactive Controller as a fallback module. During the period it controls the robot, the Failure Prediction Module continues to check whether it is safe for the Data-Driven Local Controller to take back control.

3.3. Experiments and Results

Our system was verified in the CoppeliaSim simulation environment [112] that includes an omnidirectional robot platform, namely Robotino [113]. This simulation environment also provides walking pedestrian models that employ SFM for path planning. For details about the dataset used in training all deep models, including both CNPs, GAN, and RND, please refer to Appendix A. Also, tables explaining the hyperparameters used in training all networks are given in Appendix B.

3.3.1. Analysis of the Generated Global Trajectories

The application of neural networks in global planning tasks is an established procedure. Many studies in the literature propose variants of neural networks that optimize for the shortest or the minimum-energy trajectories [114]. In this part, we aim to emphasize the capability of our structure over the standard neural network approach from the perspective of social navigation.

On the dataset of trajectories described in Appendix A, we trained our Data-Driven Global Controller and a standard feed-forward neural network (FFNN) with five layers for comparison. After training, both networks were queried in novel environments with different starting and goal positions and with different pedestrian positions. A subset of the trajectories generated by both networks is given in Figure 3.7.

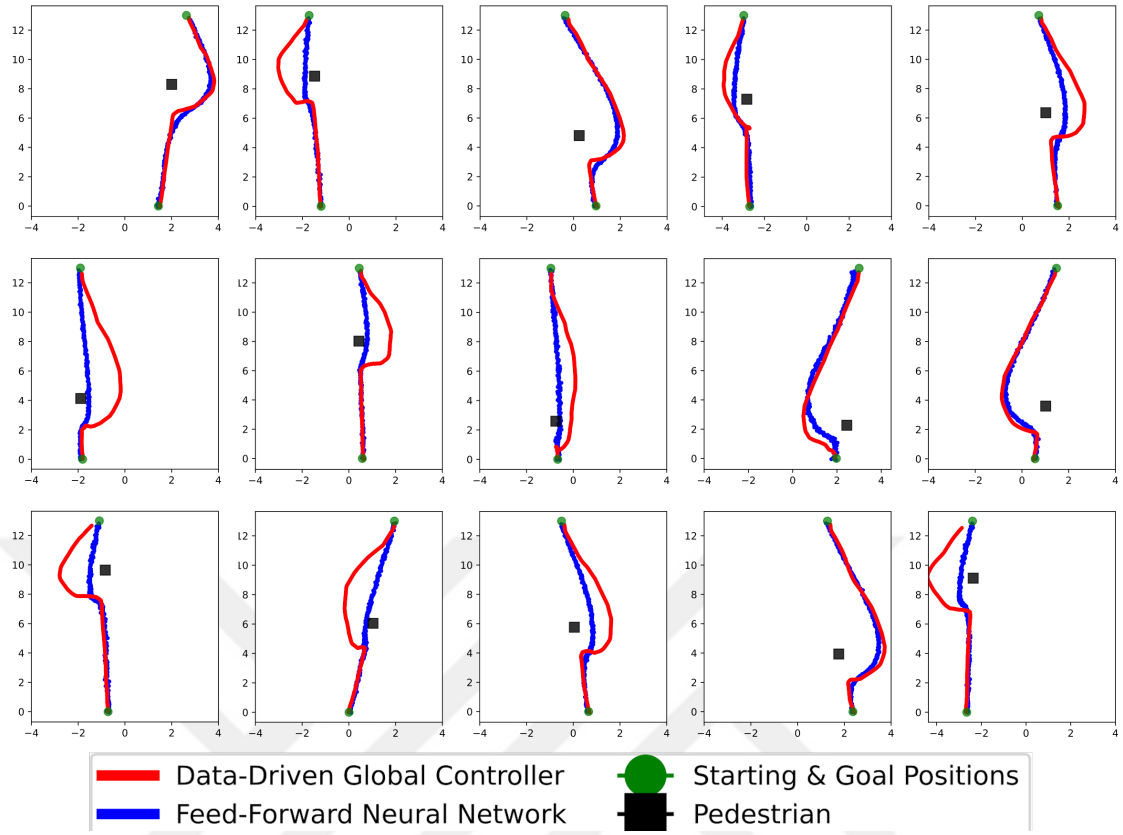


Figure 3.7. The comparison of trajectories generated by the Data-Driven Global Controller and a feed-forward neural network on several environment configurations. Results show that the Data-Driven Global Controller produces less disturbing trajectories for pedestrians.

In most cases, both approaches could generate trajectories that allow the robot to avoid pedestrians, which are shown with black dots. In many cases, though, the FFNN failed to generate global paths that avoid the pedestrian in a socially compliant manner, whereas our system could. We believe that the difference in the performance was due to the inability of standard FFNNs to encode multiple modes of operations, in other words, multiple trajectories for the same or very similar environments. In detail, given multiple trajectories that avoid the same pedestrian from different sides, our system can encode different modes in its robust latent space. In contrast, standard FFNNs learn an average representation from multiple trajectories in the same environment. Therefore, the produced trajectory by FFNNs corresponded to the average trajectory of the demonstrated ones.

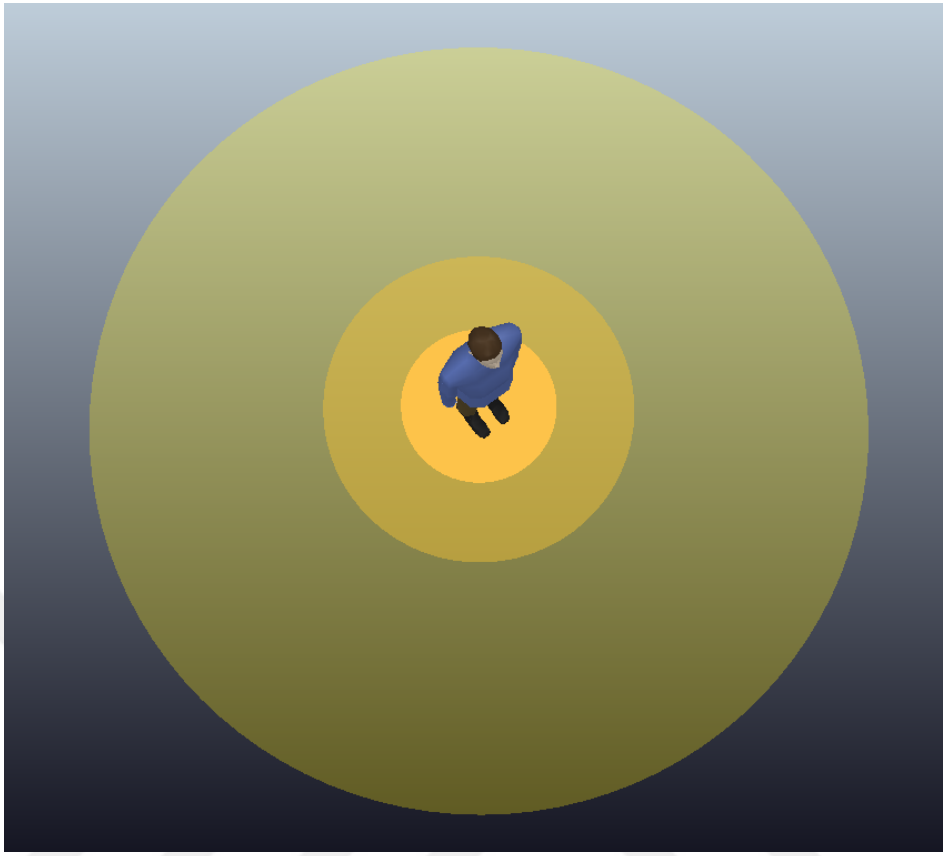
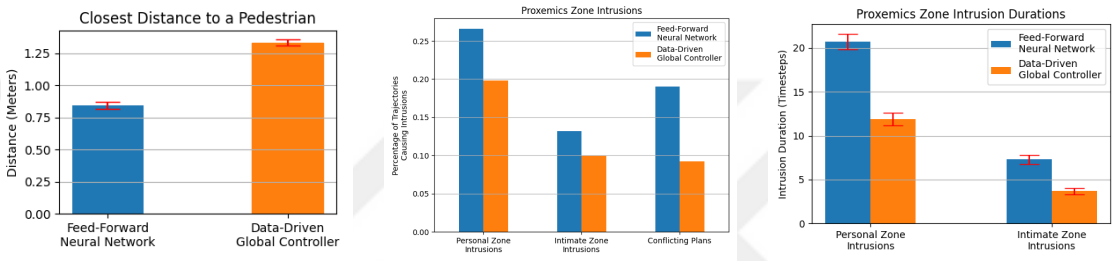


Figure 3.8. Illustration of the proxemics zones. The theory of proxemics defines four notional zones surrounding a human. These zones affect the behavior of the individual by influencing the type and the content of the interaction. These zones (from inner to outer) are Intimate (0-0.5 m.), Personal (0.5-1 m.), Social (1-4 m.), and Public (4-8 m).

In order to further analyze and compare these approaches from the social navigation perspective, we conducted a metrical comparison using the concept of proxemics. Defined by [51], the theory of proxemics investigates the spatial aspects of nonverbal communication. It suggests that interpersonal spaces are critical determinants of social interactions. In addition, there are four abstract zones surrounding a human. An illustration of them is given in Figure 3.8. The physical intrusion of these zones is only allowed under certain circumstances; for example, people can comfortably allow their family and close friends in their personal zones. On the contrary, disallowed intrusion of these zones usually leads to discomfort or anxiety.

We present the metrical comparison of these approaches in Figure 3.9. Using 1000 randomly generated environments, we first measure the closest distance of the generated trajectories to a pedestrian in the environment and compare the methods based on the closest distances. The Data-Driven Global Controller is better at producing trajectories that do not come too close to pedestrians. The closer the distance to a pedestrian, the more disturbing the robot becomes [31].



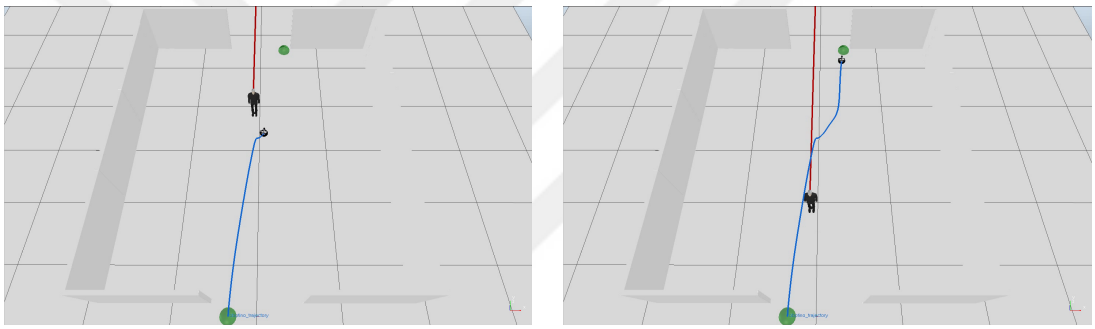
(a) (b) (c)

Figure 3.9. (a) shows the closest distances to a pedestrian both approaches produce. (b) and (c) show counts and duration of zone intrusions. Data-Driven Global Controller passes close to the pedestrian less frequently, causing less discomfort. Proxemics-zone intrusions support this argument. Refer to the text for more details.

Similar reasoning is used in [29], where researchers created a hand-crafted system to increase the lateral distance to the closest pedestrian that the robot leaves during the passing behavior. In [63], researchers also use the closest distance as a metric to evaluate their IRL-based approach. Moreover, [62] uses a distance-based comfort metric to compare navigation approaches from the social navigation perspective: the number of intrusions of the intimate and personal zones. In addition to that, we also use the duration of zone intrusions as an indicator of discomfort. A more extended period of intrusion of proxemics zones naturally leads to a higher level of anxiety. As shown in Figure 3.9, our approach performs substantially better than the standard FFNN approach in all cases. The differences between the two approaches in both closest-distance and zone-intrusion values are statistically significant upon the application of the t-test and z-test with a confidence level of 0.95.

3.3.2. Analysis of the Local Controller: Evasive Maneuvers

Our model makes instantaneous action decisions at the local controller level to respond to the changes in the environment. For convenience, the model considers only the closest pedestrian, but the underlying neural network can handle multiple pedestrians. When it is inevitable to diverge from a straight path due to a possible conflict with a pedestrian, the local controller navigates the robot safely, following the norms taught by the expert demonstrations. To establish these claims, the performance of the trained local controller is assessed on a set of simulated tasks.

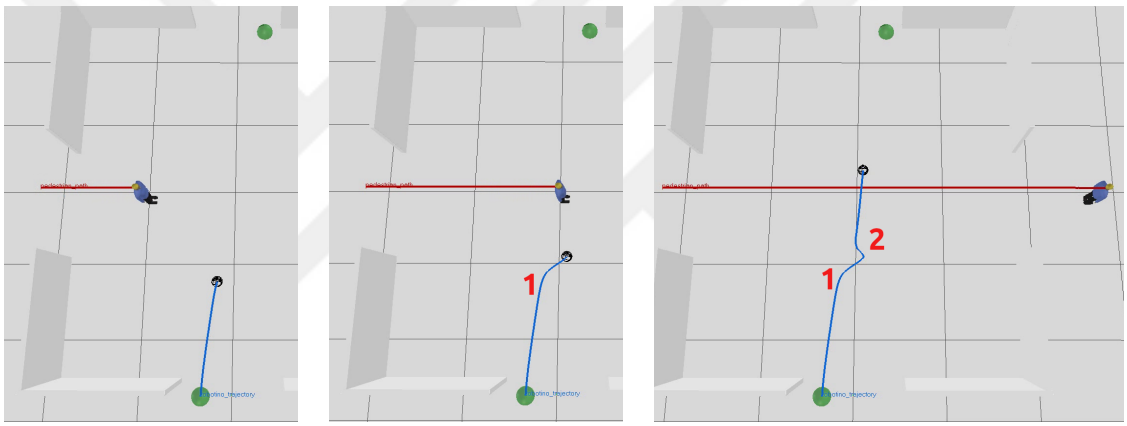


(a) (b)

Figure 3.10. Snapshots from the scene where the pedestrian moves on a vertical trajectory shown in red. The robot, controlled by the Data-Driven Local Controller, aims to reach its target, shown with a green dot at the top. When it encounters the pedestrian (a), it avoids her, reactively changing its trajectory (b), as shown in blue.

Firstly, the obstacle avoidance capability of the Data-Driven Local Controller was tested in configurations with a moving pedestrian. Although the training dataset does not contain any scenarios with moving pedestrians, our local controller could generalize to such cases. In Figure 3.10, the robot alters its trajectory, shown in blue color, dynamically to steer away from the pedestrian, whose trajectory is shown in red color. The execution of the same motion primitive is more apparent in Figure 3.11. In this case, the robot maneuvers multiple times to avoid any probable zone intrusions while moving toward its goal.

Second, we tested the Data-Driven Local Controller in a scenario where multiple pedestrians were situated. Although the controller is trained in environments with single and stationary pedestrians, it can scale this behavior into more crowded environments by considering only the closest pedestrian at test time. In Figure 3.12, the social aspect of the controller is rather apparent. Instead of going towards the goal directly, which is efficient and physically safe, the robot exhibits a social behavior of steering away from the pedestrians as its priority. Since such a norm was present in the expert behavior, the Data-Driven Local Controller captures this as a motion primitive.



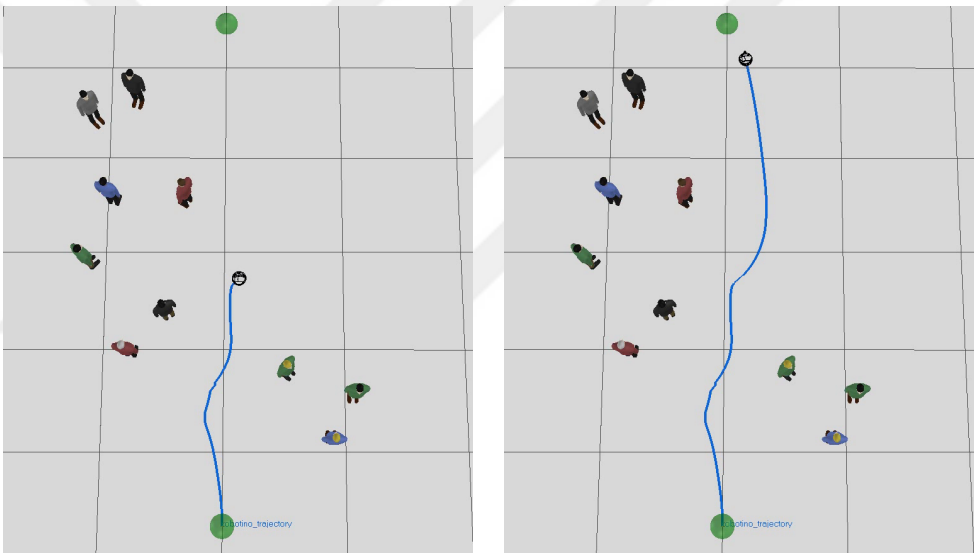
(a) (b) (c)

Figure 3.11. The pedestrian is moving on a horizontal trajectory, shown with red, and the trajectory of the robot is shown with blue. In (a), the robot aims to reach its target, shown at the top with the green sphere. In (b), the robot alters its trajectory first to evade the pedestrian on its left. In (c), as the pedestrian continues to move, the robot makes another maneuver to evade the pedestrian on its right.

3.3.3. Comparison of Local Controllers

To assess the success of our system, we devised a set of 25 simulated tests with three local controllers; Social Force Model, CNP trained on Simulation, and CNP trained on SCAND. SCAND [80] is a real-world dataset gathered on indoor and outdoor social environments with human demonstrators. More details on the dataset are

given in Appendix A. The quantitative comparison is given in Table 3.1. Average Displacement Error (ADE) shows how much the robot steers away from the straight line from the start position to the goal position. Path Length (PL) is used to measure the length of the generated trajectories, whereas Average Time to Goal (ATG) shows how long it takes for the robot to get to the goal position. Average Acceleration (AA) is reported by summing up the changes in the acceleration throughout the navigation task. Proxemics Zone Intrusion Counts (PZIC) measure the number of personal zone intrusions. SI units are used in the comparison.



(a) (b)

Figure 3.12. Snapshots from the scene where the robot is passing through several pedestrians. Instead of going directly to its target, the robot takes a longer trajectory (shown in blue), complying with the social norms present in the dataset. This behavior decreases proxemics-zone intrusions.

Several quantitative metrics are used to compare the three controllers. Average Displacement Error (ADE) and Average Acceleration (AA) metrics are proposed in [72], whereas Path Length (PL) and Average Time to Goal (ATG) in [115]. Proxemics Zone Intrusion Counts (PZIC) metric is proposed in [62].

According to these tests, the most notable difference is the velocity each agent prefers in the same condition. Demonstrators seem to prefer lower speeds around people, whereas our simulated controller and the one trained on this data travel at higher speeds. Therefore, even though PL values are very close, the controller trained on real-world data scores substantially higher ATG values. This situation is also reflected in the AA dimension. The controller trained on real-world data tends to change its speed less frequently than the other two. On the other hand, this controller makes more personal zone intrusions, signaling that human demonstrators prefer slowly diverging from other pedestrians' paths in case of encounters.

Table 3.1. Comparison of SFM and CNP approaches.

	Social Force Model	**CNP on Simulation**	**CNP on SCAND**
ADE	0.43 ± 0.01	0.53 ± 0.01	0.72 ± 0.02
PL	13.18 ± 0.01	13.49 ± 0.03	13.10 ± 0.01
ATG	22.83 ± 0.08	34.83 ± 0.38	69.86 ± 0.80
AA	5.24 ± 1.51	6.51 ± 0.07	-0.06 ± 0.07
PZIC	0	0	3

3.3.4. Performance of the Complete System

Finally, we demonstrate the contribution of our pipeline in terms of social navigation with a quantitative comparison. In the same environment as shown in Figure 3.12, we randomly distributed the pedestrians and compared the executed navigation trajectories of the proposed system and a traditional navigation system. The traditional planner implements the A* algorithm at the global level [116] and the Elastic Bands approach at the local level [117]. Figure 3.13 summarizes the numerical analysis. In Figure 3.13(a), we compare both methods in terms of the closest distances they approach a pedestrian. Closer distances negatively affect the comfort levels of pedestri-

ans [31]. The comparison shows that our approach leads to less discomfort by leaving more space with the surrounding people. In Figure 3.13(b), we present proxemics-zone intrusion counts for both methods, and in Figure 3.13(c), zone-intrusion durations are shown. These comparisons also show that our pipeline makes fewer and more brief intrusions of the proxemic zones of pedestrians. Lastly, in terms of distance-based metrics, these differences are statistically significant upon the application of the t-test, with a confidence level of 0.95.

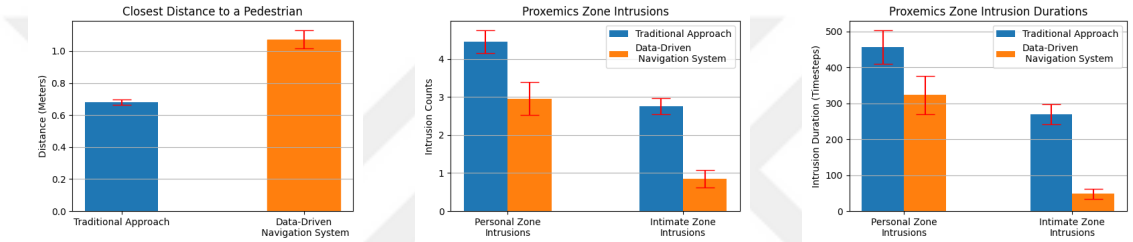


Figure 3.13. In (a), we show the closest distances to a pedestrian both approaches produce. Our pipeline minimizes the discomfort of pedestrians by staying distant from them during navigation. This is also reflected in (b) and (c), where we compare two methods in terms of zone-intrusion metrics. The count and the duration of zone intrusions are significantly less for our system. Refer to the text for more details.

3.3.5. Contribution of the Failure Prediction Module

The Failure Prediction Module that we incorporated into our navigation system aims to recognize potential extrapolating cases before any collision occurs. In the following, we first illustrate the contribution of this module. Later, we present an ablation study where we measure the performance of the complete pipeline with and without the Failure Prediction Module.

Figure 3.14(a) shows a failed extrapolation of the data-driven modules, which leads to a collision. The Failure Prediction Module detects such extrapolation cases and enables the navigation system to take measures, as shown in Figure 3.14(b).

(a) (b)

Figure 3.14. Navigation system with and without the Failure Prediction Module in the same environment. (a) shows when the robot relies only on the data-driven modules of the navigation system. In (b), the Failure Prediction Module detects collisions in advance and gives control temporarily to the Hand-Crafted Reactive Controller, enabling the robot to move safely within the cluttered area.

In order to demonstrate the contribution of the Failure Prediction Module to the Data-Driven Navigation System, we compare the performance of the entire navigation system with and without this module. Also, we present a comparison between the two outlier detection approaches on our system, namely the GAN and the RND. We ran 50 tests with randomly positioned pedestrians in the environment, just like the one shown in Figure 3.14. The numbers of collisions and proxemics-zone intrusions for three cases are shown in Figure 3.15.

Extrapolation errors occurred in 14 tests, and without the Failure Prediction Module, such errors eventually led to collisions with pedestrians. On the other hand, the addition of the Failure Prediction Module enabled the successful prediction of

all extrapolating cases, with the exception of one case where RND noticed the extrapolation too late for the robot to recover. Upon the prediction of extrapolation, the Hand-Crafted Reactive Controller Module temporarily took control of the mobile base, avoiding collisions altogether in all instances. However, results show that the numbers and durations of the proxemics-zone intrusions are higher for the system with the Failure Prediction Module. There are two reasons for this situation. First, tests are finished early when a collision occurs; the robot could simply not complete tests with collisions. Therefore, it was not possible to examine the performance in the rest of the task leading to a low number of intrusions. Second, although the system with the Failure Prediction Module successfully executed social navigation in most cluttered cases, some of these environments were just too cluttered. In these cases, the navigation pipeline eluded collisions by turning them into zone intrusions.

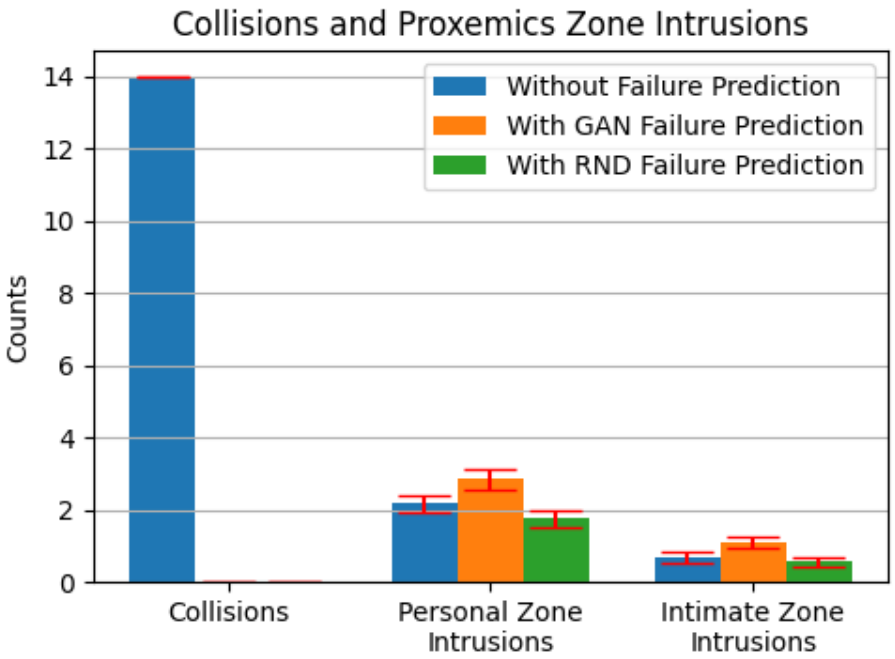


Figure 3.15. Performance of the Data-Driven Navigation System with and without the Failure Prediction Module in different environments with randomly placed pedestrians. Integration of the Failure Prediction Module successfully resolves collisions in each case. Refer to the text for more details.

Table 3.2. Effect of the size of the demonstrations set on model performance.

	Collision Count	**ADE**	**IZIC**	**PZIC**
D = 50	5	0.17 ± 0.02	5	5
D = 250	4	0.3 ± 0.03	5	6
D = 500	0	1.12 ± 0.02	5	10
D = 1000	0	0.25 ± 0.02	4	7
D = 1500	0	0.96 ± 0.02	0	0

3.3.6. Scalability of CNP

To evaluate the scalability of our controllers, we train the Data-Driven Local Controller Module with datasets of different sizes and perform simulated tests with the learned controllers. The results are given in Table 3.2. Here, ADE refers to Average Displacement Error, while IZIC and PZIC refer to Intimate and Personal Zone Intrusion Counts. According to these tests, when the dataset size is small, when the size is D=50 or D=250, we see that the controller struggles to avoid collisions with pedestrians. The controller starts to learn meaningful avoidance behaviors when D $\geq$ 500. As D increases, we see a decrease in the proxemics-zone intrusion counts. Average Displacement Error reaches meaningful levels only when D $>$ 1000.

3.4. Conclusion

This study presents a novel navigation system that can learn local and global navigation directly from expert demonstrations. We used a state-of-the-art deep learning framework, namely Conditional Neural Processes (CNPs), that can learn multimodal distributions around complex trajectories from relatively smaller datasets compared to its alternatives. One common problem with data-driven systems is that they are prone to extrapolation errors. Therefore, using only the data-driven architectures would eventually lead our system into collisions. To minimize this possibility, we devise a

layered architecture inspired by the subsumption architecture [118]. We leverage state-of-the-art deep learning frameworks, namely Random Network Distillation (RND) and Generative Adversarial Networks (GANs), to work as arbitrators in this hierarchy. We have compared their performances and found that RND works better on average to detect out-of-distribution (OOD) samples enabling the system to infer the situations leading to extrapolation errors. When OOD states are encountered, the control of the robot is temporarily given to a manually encoded controller. We use an SFM-based controller for this purpose, but any socially-aware manually-encoded controller can be employed in the Hand-Crafted Reactive Controller.

In a simulated environment with a mobile robot, stationary and moving pedestrians, and given target points, we showed that our method could generate socially aware paths at the global level and successfully avoid pedestrians at the local level. This idea opposes the conventional approach of handling social navigation only in the local layer. An increase in the socialness of global trajectories improves the overall success of the social navigation frameworks. The results were verified in different simulated environments using metrics such as the average acceleration, path length, average displacement error, the closest distance to pedestrians, and intrusion amount of their proxemics zones. These promising findings should be supported by real-world tests as the reactions of the simulated people do not align perfectly with actual humans in natural environments. On the other hand, there are certain limitations to the current status of the system. An important limitation of this work is that the model does not consider the orientation and the velocity of the pedestrians. These factors would definitely alter the global and local plans of the robot. In future work, we plan to include these parameters in our models. Moreover, since we applied interpolation on the demonstration trajectories to get a continuous path, we lost the timing aspect of the trajectory in the Data-Driven Global Controller Module. We are planning to solve this issue by adopting a different representation that enables us to learn global trajectories from real-world data without interpolation so that we can produce more flexible global trajectories. There are recent benchmarking tools in the social robot navigation domain, such as [32,119,120]. They provide an automated evaluation of the algorithms

and comparison with several conventional approaches, such as SFM and RVO [121]. We plan to use these tools to automate and expedite benchmarking efforts.

We believe that the social aspects of our approach can further benefit from the integration trajectory forecasting mechanisms, such as [66]. In the future, we plan to use future predictions about pedestrian trajectories as input to the trajectory calculation of the robot to create less disturbing trajectories by increasing the preference for uncongested parts of the environment, avoiding probable future encounters. Our system uses CNPs in both global and local controller modules. On the other hand, these two modules are independent of each other. We believe we can leverage the capability of CNPs in both layers, but some different approaches, like LSTMs or transformers, can also be used here. We also aim to investigate such techniques in the global controller layer in the future. Moreover, we aim to deploy this system on real robots. Since robots in the real world need to be controlled in real time for successful navigation, the challenge in applying our system is obtaining high-level parameters that we feed to CNPs in real time. We believe that this can be addressed by leveraging the flexibility of CNPs to learn representations from any low-level and high-dimensional input, as shown in [122].

4. CONDITIONAL NEURAL EXPERT PROCESSES FOR LEARNING MOVEMENT PRIMITIVES FROM DEMONSTRATIONS

4.1. Introduction

In the absence of a formal definition of what social is, one practical way to create social robotic movements is to directly teach them these behaviors. Learning from Demonstration (LfD) is a widely adopted procedure in robotics that enables learning controllers to acquire new skills by observing an expert [7, 123]. For this purpose, elaborate demonstrations of target skills are required in LfD. On the other hand, the set of expert demonstrations for a particular real-world skill may contain significant variances, or there might be multiple ways to achieve the same skill. These variances reflect the stochastic nature of the expert demonstrations, which poses the challenge of handling quantitatively and qualitatively different demonstrations for LfD-based skill-acquisition procedures.

Assume a human is given the task of teaching a robot a diverse set of skills, all initiated in the same environment setup. Given a large number of demonstrations without any further supervision about the type of motion, it is challenging to train a single system to handle such diversity. Instead, as Schaal et al. stated, "the existence of movement primitives seems, so far, the only possibility how one could conceive that autonomous systems can cope with the complexity of motor control and motor learning" [10]. Using this insight to address the challenges caused by variances of demonstrations, we aim to develop a system that autonomously discovers the movement primitives while learning to generate the corresponding sensorimotor trajectories.

In this section, we introduce the proposed Conditional Neural Movement Primitives (CNEP) model, a generic and monolithic LfD framework to teach robots the necessary controllers to model and synthesize complex, multimodal sensorimotor tra-

jectories. In previous approaches, such as [10, 16, 41, 91], the multimodality aspect of target skills was not explicitly addressed as these approaches attempted to represent different modes with the same mechanism, leading to a seamless interpolation inside the demonstration space, which may lead to suboptimal behavior, as shown in [43]. On the contrary, our CNEP is designed to model different modes in the demonstrations with different experts and generate the required motion trajectory by automatically selecting the corresponding expert.

Our model builds on top of Conditional Neural Movement Primitives (CNMP) [16], which was shown to form robust representations to model complex motion trajectories from a few data points. CNMPs have an encoder-decoder structure that allows them to generate motion trajectories given a set of conditioning (observation) points. CNEP uses multiple decoders (experts) - instead of a single one - that are responsible for different modes in the given trajectories. Given the conditioning points and the output of the encoder network, a novel gating mechanism assigns probabilities to the experts, and the decoder with the highest probability is used to generate the motion trajectory from the encoded conditioning points. Besides the architectural contribution that includes the gating mechanism and multiple experts, we propose a novel loss function to ensure that all the experts are evenly utilized and an expert, when assigned, is selected with high probability.

4.2. Method

4.2.1. Problem Formulation

The skill-acquisition problem can be formulated as finding a sequence of motion commands that produce the desired movement [124]. Formally, the LfD system is expected to learn a function τ = f(t; X), where X denotes specific criteria, such as the starting point at t=0 or the destination at any time t, using N expert demonstrations, D = $\{\tau_1, \tau_2, \ldots, \tau_N\}$. Despite the multimodality of the target skill, a resource-efficient solution with few demonstrations is also demanded to promote the applicability of

the proposed approach in real-world settings where it is infeasible to provide so many demonstrations.

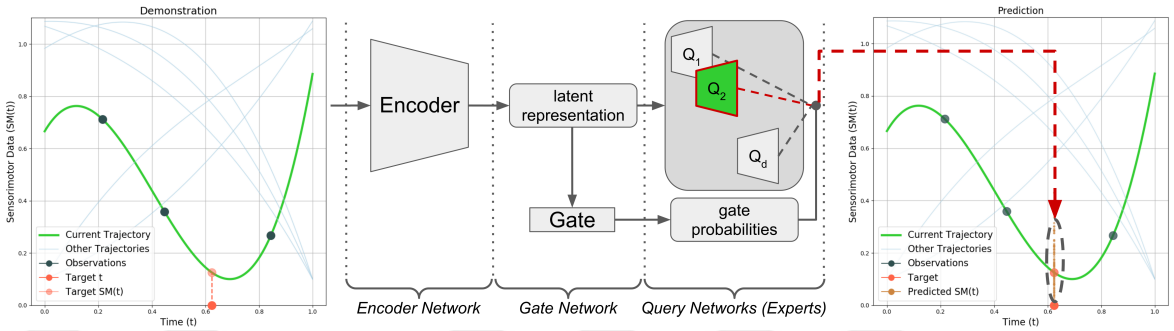


Figure 4.1. The CNEP model contains an Encoder, a Gate, and multiple Query Networks called experts. In this example, n=3 observations (shown with •) and m=1 targets (shown with •) are randomly sampled on the input. The latent representation is used by the Gate Network to find the responsible expert. Please, refer to Section 4.2.3.1 for more details on the operation.

In this context, sensorimotor functions (SM(t)) are utilized to refer to the temporal mapping of sensory inputs and motor outputs of a robot at time t. Two important notions are encapsulated in the SM(t) formalism: (1) how a robot senses its environment through sensors and (2) how it responds through actuators at any given moment. The perspective of representing complex skills as SM(t) trajectories transforms skill-acquisition efforts into trajectory modeling and generation problems. A trajectory is formally defined as a temporal function, $\tau = \tau(t)$ following [125]. Throughout this study, each trajectory τ is represented as an ordered list of sensorimotor values: $\tau = \{SM(t_1), SM(t_2), \ldots, SM(t_T)\}$.

In the following section, after introducing the baseline (CNMP) method, we provide details of our proposed method (CNEP). As LfD frameworks, both are used to encode a set of trajectories from expert demonstrations. They take a set of observation (conditioning) points, in the form of (t, SM(t)) tuples from a trajectory, and are expected to output the SM(t_q) value of any target timepoint, t_q. In practice, given

varying observation points, the entire trajectory is generated by querying the system for all time points from t_1 to t_T.

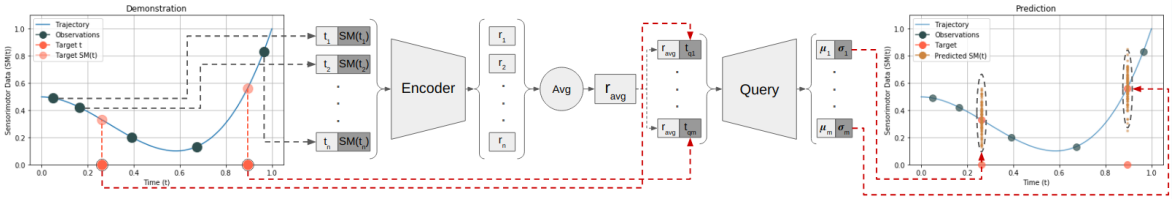


Figure 4.2. Operation of the CNMP model. The Encoder network encodes observations. Latent encodings are averaged to generate a compact representation of the entire trajectory. Based on the average latent representation (r_{avg}), the Query Network outputs its prediction about the sensorimotor responses at query times.

4.2.2. Background: CNMP

Introduced in [16], the CNMP model is a variant of the Conditional Neural Processes (CNPs) [41] aiming to model and generate robotic skills in terms of movement primitives. As depicted in Figure 4.2, the model contains an Encoder and a Query Network. At the training time, a trajectory τ_i is first sampled from demonstrations, D. n randomly sampled observation points from this trajectory are passed through the Encoder Network to generate corresponding latent representations. An averaging operation is applied to obtain a compact representation of the (n) input observations in the latent space. The average representation is then concatenated with m random target time points and passed through the Query Network to output the distributions that describe the sensorimotor responses of the system at corresponding target time points. Here, n and m are random numbers, where $1 \leq n \leq n_{max}$ and $1 \leq m \leq m_{max}$. n_{max} and m_{max} are hyperparameters whose values are set empirically. The output is a multivariate normal distribution with parameters (μ_q, Σ_q). The loss is calculated as the negative log-likelihood of the ground-truth value under the predicted distribution

$$\mathcal{L} = -\log P(\, SM(t_q) \,|\, \mathcal{N}(\mu_q,\ softplus(\Sigma_q)\,)\,), \tag{4.1}$$

which is backpropagated through the entire model. The derivative of the backpropagated loss is used to adjust the weights of both the Encoder and the Query networks. More details can be found in [16].

4.2.3. Proposed Approach: CNEP

4.2.3.1. Architecture Overview. The proposed architecture and the workflow are illustrated in Figure 4.1. Similar to the CNMP, the input to the system is composed of n observations, which are passed through the Encoder Network to generate the latent representation. The latent representation is fed into our novel Gate Network to produce the gate probabilities for all experts. Simultaneously, they are concatenated with m target time points and are passed through all experts to generate SM predictions at target time points. During training, the Gate Network's output is combined with the experts' outputs to compute the overall loss. After training, when the system is asked to generate a response for a target, only the prediction of the expert with the highest gate probability is outputted. An example case is shown in Figure 4.1, where after producing the latent representation for n=3 observation points, the gating mechanism outputs a relatively high probability for the second expert, appointing it as the responsible expert for this query. Therefore, its response for m=1 target time point is selected as the output of the entire system. The system is trained end-to-end, as detailed in the next section.

4.2.3.2. Training Procedure. The training phase of the CNEP model is depicted in Figure 4.3. From a randomly selected trajectory, τ_i, n observation points are randomly sampled to form (t, SM(t)) tuples. Once the input containing a set of (t, SM(t)) tuples is obtained, it is passed through the Encoder Network. A compact representation estimate for τ_i, r_i, is obtained by averaging the output of the Encoder Network for the n conditioning points in the latent space. In Figure 4.3, parallel processing of a batch of trajectories is shown.

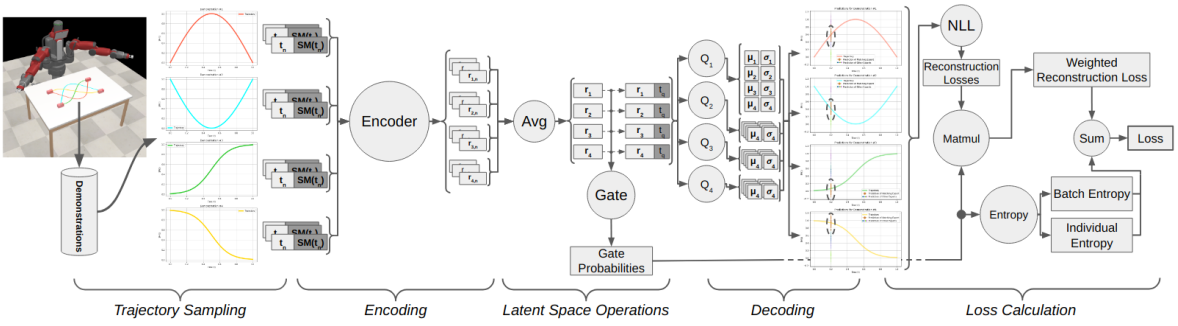


Figure 4.3. Operation of the CNEP model. First, observation points are mapped to the latent space by the Encoder. The averaged representations (r) are: (1) fed into the Gate Network and (2) concatenated with target points (r_q) and fed into the Query Networks. While all Query Networks output the candidate predictions, the probabilities for each trajectory-expert pair (p) are generated by the Gate Network.

The training procedure comprises three crucial steps, which are detailed in the following:

(i) Calculating gate probabilities: The gate probability is the probability that the set of conditioning points and their underlying trajectory will be generated by the corresponding expert, as predicted by our system. For each expert e, the gate probability $p_{i,e}$ is calculated by the Gate Network, which takes as input the average latent activation (r_i) of the corresponding conditioning points and passes it through a linear layer followed by the Softmax function. The number of experts, d, is a hyperparameter in our model.

(ii) The reconstruction loss: Each expert predicts a normal distribution for the SM values at target time points, and a reconstruction loss is calculated using the ground-truth SM values. For this, m target timepoints (t_q) are randomly sampled, where m is set to 1 in Figure 4.3 for simplicity. A (r_i, t_q) tuple is passed through all experts (Q_es) to generate their predictions. The negative log-likelihood of the actual SM(t_q) under the predicted distribution is computed as the reconstruction loss of the corresponding expert by

$$\mathcal{L}_e = -\,log\ P(\ SM(\mathbf{t_q})\,|\,\mathcal{N}(\mu_e,\ softplus(\mathbf{\Sigma_e})\ )\ ),$$

where μ_e and Σ_e are the outputs of the Query Network, Q_e. Next, the combined

reconstruction loss for the trajectory τ_i is calculated by taking the weighted sum of L_es of all experts using

$$\mathcal{L}_{rec}^{i} = \frac{1}{d}\sum_{e=1}^{d} L_e \times p_{i,e},$$

where $p_{i,e}$ is the probability of expert e for the latent representation r_i. For a batch of trajectories (Figure 4.3), the weighted reconstruction loss $\mathcal{L}_{rec}$ is the mean $\mathcal{L}_{rec}^{i}$, where $1 \leq i \leq b$, and b is the batch size.

(iii) Expert assignment losses: We would like our system to avoid selecting the same expert for all possible modes. For this purpose, the entropy of the expert activation frequencies over a batch of training trajectories should be maximized. We use batch entropy ($\mathcal{L}_{\text{batch}}$) to enforce this constraint. In the meantime, we would like the system to attribute a high probability when assigning a (r_i, t_q) tuple to one of the experts. For this, the entropy of the gate probabilities, p_i, should be minimized. We use individual entropy ($\mathcal{L}_{ind}$) to enforce this constraint.

Formally, $\mathcal{L}_{\text{batch}} \in \mathbb{R}$ is calculated by

$$\mathcal{L}_{\text{batch}} = -\sum_{e=1}^{d}\left(\frac{1}{b}\sum_{i=1}^{b} p_{i,e}\right)\log\left(\frac{1}{b}\sum_{i=1}^{b} p_{i,e}\right),$$

where, again, b is the batch size, and d is the number of experts. Subsequently, $\mathcal{L}_{ind} \in \mathbb{R}$ is computed using

$$\mathcal{L}_{ind} = \frac{1}{b}\sum_{i=1}^{b}\left(-\sum_{e=1}^{d} p_{i,e}\log(p_{i,e})\right).$$

The entire system is trained in an end-to-end manner where the parameters of the Encoder, the Gate, and the Query Networks are trained simultaneously in a supervised way. The overall loss function is a linear combination of the abovementioned three components: (1) the weighted reconstruction loss, (2) batch-wise expert activation loss, the batch entropy, and 3) trajectory-wise expert selection probability, the individual entropy. The dynamic nature of expert selection necessitates careful handling during training to achieve the right balance between expert adaptation and overall system stability. Essentially, as the model attempts to decrease the reconstruction loss, it stimulates experts to make accurate predictions. Moreover, while the model attempts to increase batch entropy, it promotes expert specialization by preventing the overuti-

lization of any expert. Lastly, the individual entropy component of the loss indirectly implies the confidence of latent representation-expert matching. As the model attempts to decrease this value, it contributes to the specialization of experts by assigning similar representations to the same expert. As a result, the overall loss is calculated as shown in Figure 4.4, which is formally given as

$$\mathcal{L}_{total} = \alpha_1 \times \mathcal{L}_{rec} + \alpha_2 \times \mathcal{L}_{batch} + \alpha_3 \times \mathcal{L}_{ind}, \tag{4.2}$$

where weighting coefficients of these components, α_1, α_2, and α_3, are found empirically by the grid search technique using a specific library, called Weights & Biases [126]. Eventually, the loss calculated by Equation(4.2) is backpropagated through the model.

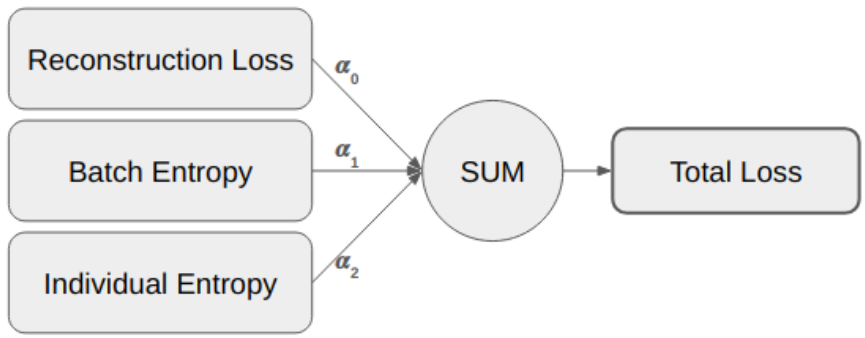


Figure 4.4. Overall loss of the CNEP model. To simultaneously optimize trajectory reconstruction and in-batch expert utilization and to promote expert specialization, a linear combination of these three different loss components is computed and backpropagated throughout the entire system.

4.2.3.3. PID Controller . To execute learned skills on the robot, a PID controller is appended to the end of our system, ensuring that synthesized trajectories pass through specified observation points. The predicted SM trajectories are fed into this controller prior to the execution. The control signal $\mathbf{u}$(t) for each timestep is calculated using the PID formula

$$\mathbf{u}(t) = K_p \cdot \mathbf{e}(t) + K_i \cdot \int \mathbf{e}(t)\, dt + K_d \cdot \frac{d}{dt}\mathbf{e}(t)$$

where $\mathbf{e}(t)$ is the error vector at time t, and K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively, which define the controller's behavior.

The error $\mathbf{e}(t)$ is the difference between the current point and the conditioning point. A decay mechanism of certain timesteps is incorporated into the error calculation. This ensures that the corrections diminish, allowing the trajectory to converge smoothly toward the desired path.

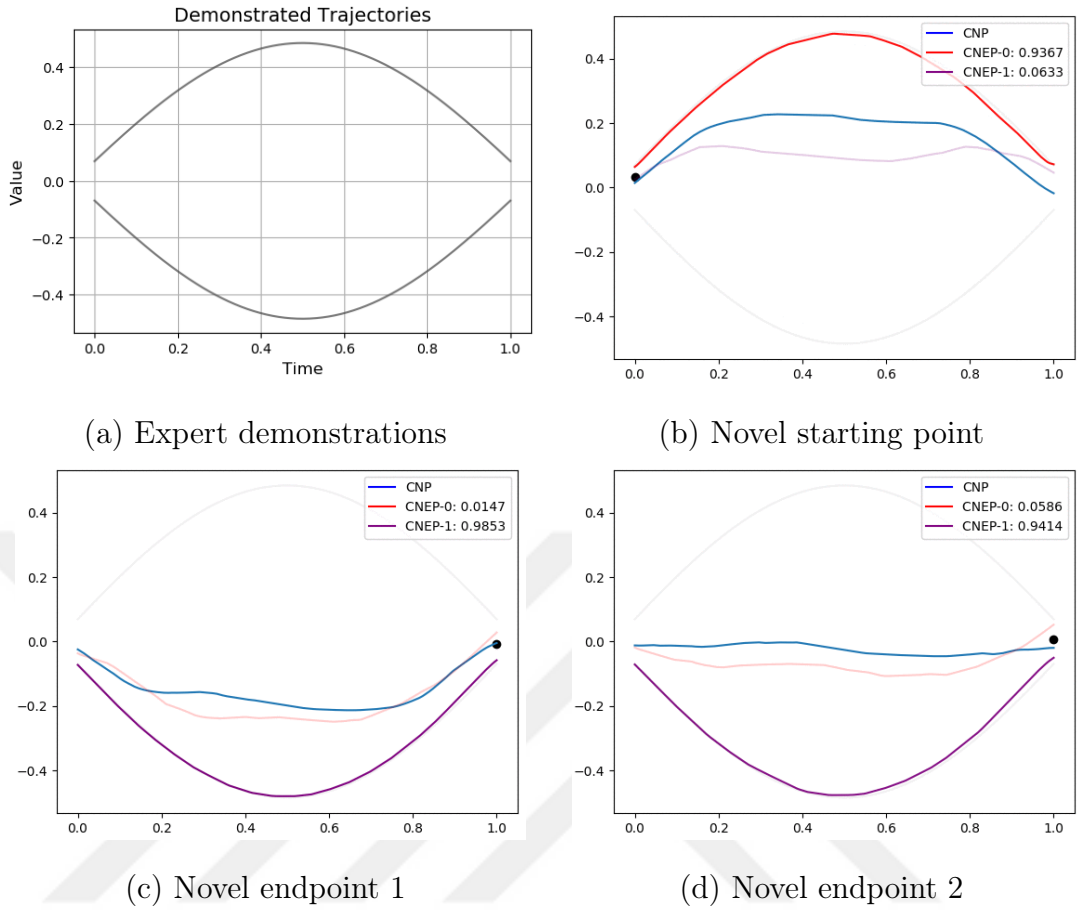
4.3. Evaluation of the Model

This section evaluates the CNEP model by showing its performance in learning movement primitives through multiple experiments. The first set of tests uses synthetically produced trajectories as sensorimotor demonstrations and thoroughly assesses the model against an ensemble of movement primitive-based approaches. Furthermore, two ablation studies are provided to exhibit the effects of individual components of the model. Experiments with the simulated humanoid model show the ability of the CNEP model to process high-dimensional sensory data and produce high-dimensional control commands. Finally, the proposed approach is evaluated on two real-world tasks with a real robotic manipulator. These final experiments demonstrate the applicability of the CNEP model in practical cases.

4.4. Experiments and Results

4.4.1. Robotic Skills with Multimodal Sensorimotor Trajectories

To expose the main advantage provided by the CNEP model, a test in a simplistic setting with only two expert demonstrations is conducted. Given in Figure 4.5, assume that a robotic skill can be realized either by the trajectory above or the one below. This illustrative comparison is encountered frequently in LfD settings. When the preference of the demonstrator is the mere determinant of the selection between two candidate solutions, a bimodal solution space is obtained. This scenario is a simplified version of the real-robot experiment, explained in Section 4.4.5.1.



(a) Expert demonstrations (b) Novel starting point

(c) Novel endpoint 1 (d) Novel endpoint 2

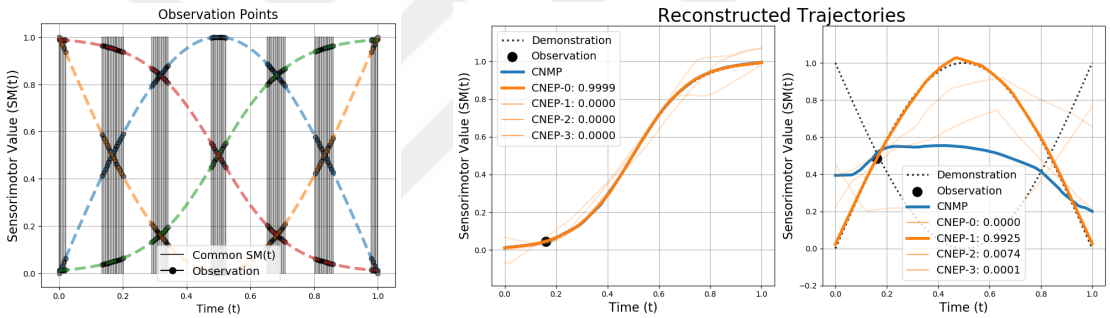
Figure 4.5. Comparison of trajectory generation performances of CNP and CNEP models against a set of multimodal expert demonstrations.

Many Gaussian-based approaches, such as the original CNP [41], attempt to model the demonstrations using a single distribution. Therefore, when asked to generate a novel trajectory from a novel starting point, these traditional approaches produce a mean response combining the information they are provided with. On the contrary, CNEP treats different modes in the demonstrations differently.

The rest of Figure 4.5 demonstrate example trajectories generated by CNP and CNEP models. As explained above, while CNP produces a mean trajectory when conditioned from a novel starting point, CNEP produces multiple candidate outputs and selects only the one with the highest gate probability.

4.4.2. Modelling Different MPs with Common Points

In this section, we aim to evaluate the performance of our model in a scenario where the model needs to learn from a diverse set of demonstrations with common points. CNEP and CNMP are trained with the dataset of four SM trajectories shown in Figure 4.6(a), where the trajectories intersect at various points. This figure also provides sample trajectory generations from CNEP and CNMP models. The numbers on the legend next to CNEP correspond to the probability of the assignment of one of the experts. The dashed lines correspond to the demonstration trajectories in the dataset.



(a) (b)

Figure 4.6. Assuming that the target skills to be learned contain intersecting SM trajectories, as shown in (a), it becomes challenging to distinguish them from each other. When conditioned from the observation points given in (a), CNMP outputs an average of candidate trajectories, while CNEP successfully generates the trajectory that achieves one of the target skills, as shown in (b).

As shown in Figure 4.6(b), when CNMP and CNEP are conditioned from points unique to single trajectories, both models generate the required trajectories successfully. However, when they are conditioned from points close to the intersection, CNMP starts failing, whereas CNEP successfully generates the correct trajectories, as shown in the right hand side of Figure 4.6(b). As shown, while the CNMP model generates a trajectory that resembles an interpolated trajectory, our model assigns a high probability

to one of the experts, which generates a trajectory close to one of the demonstrations in the dataset. Note that the obtained high probability value, when the conditioning point is very close to the intersection, is due to the individual entropy term in our loss function and our winner-take-all strategy.

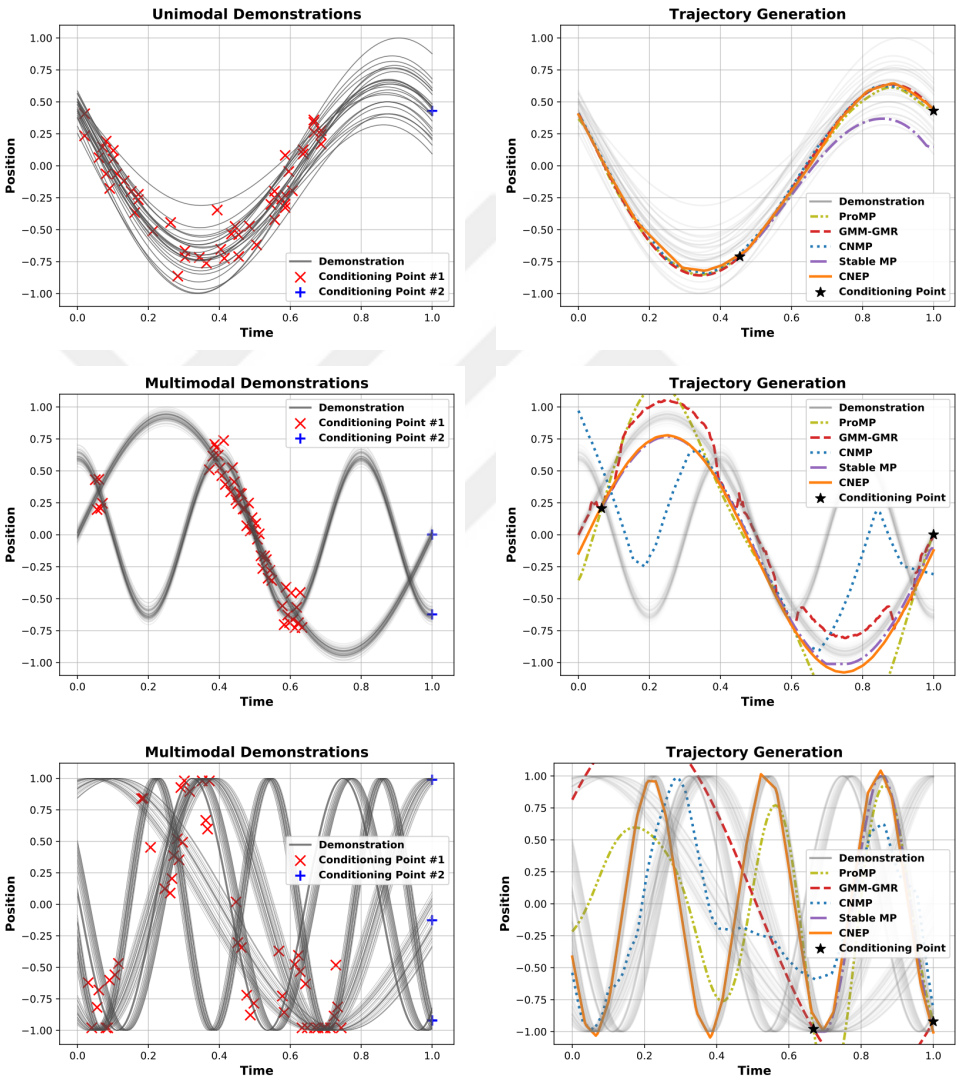


Figure 4.7. The left column presents datasets of sensorimotor trajectories with increasing complexities. Correspondingly, the right column presents synthesized trajectories on an example run upon training. Modeling the target skills becomes more challenging as the number of modalities increases. However, different experts inside the CNEP model can successfully handle the increasing complexity.

4.4.3. Comparison on Trajectories with Increasing Complexities

To highlight the advantages of the CNEP model, its performance is evaluated and compared to a group of baseline methods, including ProMP, GMM-GMR, CNMP, and Stable MP. Three datasets of sensorimotor trajectories with gradually increasing complexities are created. These datasets are shown on the left side of Figure 4.7.

After training each method with the datasets, a test set is created with 50 pairs of intermediate and end conditioning points and their corresponding ground-truth trajectories. The intermediate conditioning points were sampled from the regions where trajectories of different modes come close, as shown with the red cross (x) markers. The endpoints were sampled from the training range, as shown with the blue plus (+) markers in the same figure. The ground truth trajectories in evaluations were selected as the trajectories closest to the conditioning points in the demonstration set. The Mean-Squared Error (MSE) along the generated and ground truth trajectories were calculated for each query, and the mean and standard deviation of the errors are reported in Table 4.1.

Table 4.1. MSE over generated trajectories when conditioned from via points.

	Unimodal	**Bimodal**	**Multimodal**
ProMP	0.017 ± 0.010	0.312 ± 0.314	0.199 ± 0.191
GMM-GMR	0.017 ± 0.009	0.096 ± 0.120	0.162 ± 0.065
CNMP	0.015 ± 0.006	0.043 ± 0.078	0.112 ± 0.074
Stable MP	0.016 ± 0.014	0.017 ± 0.035	0.058 ± 0.071
CNEP	**0.015 ± 0.005**	**0.013 ± 0.013**	**0.027 ± 0.042**

Each method successfully generated plausible trajectories with low errors when the demonstration trajectories come from a unimodal distribution. However, as the complexity of the dataset increased, our method CNEP outperformed all other methods. The right side of Figure 4.7 features several sample trajectories generated by these

methods. As shown, while CNMP, GMM-GMR, and ProMP generate extrapolated, interpolated, or shifted trajectories that might correspond to mixed combinations of the demonstrated ones, our CNEP model can select the correct primitive and generate the target trajectory successfully.

4.4.3.1. Influence of the Loss Components. The loss term is one of the novel contributions of this study, and the influence of the components in the loss term requires further investigation. For this, three variants of the CNEP model were created:

(i) CNEP-Uniform (CNEP-Uni): CNEP model where the coefficient of the batch entropy term, α_2 in Equation(4.2), is set to 0. The batch entropy term promotes using all decoders to learn different skills collectively.

(ii) CNEP-UnSpecialized (CNEP-UnSpec): CNEP model where the coefficient of the individual entropy term, α_3 in Equation(4.2), is set to 0. The individual entropy term promotes decoder specialization by rewarding high-probability matches between experts and encoded representations.

(iii) CNEP-Reconstruction-only (CNEP-Rec): CNEP model where coefficients of both entropy-related loss terms, α_2 and α_3 in Equation(4.2), are set to 0. This model only considers the reconstruction loss.

In addition, our CNEP model follows a winner-take-all approach: It outputs only the prediction of the responsible expert. Another approach might be using a weighted mixture of all experts (MoE). To justify our decision and compare it with its MoE variant, a CNEP model with a Mixture-of-Experts (CNEP-MoE) has been created. These 4 variants were trained alongside the original CNEP on the datasets shown in Figure 4.7. Table 4.2 presents the MSE errors computed along the predicted and ground-truth trajectories. As shown, using all loss terms and our winner-take-all approach outperformed its variants.

4.4.3.2. Influence of the Number of Experts. The number of specialized experts inside the CNEP model also affects the overall performance. Results gathered in Table 4.3 compare 3 CNEP models with 2, 4, and 8 experts. Increasing the number of experts in the CNEP model to a number greater than or equal to the number of modalities of the demonstration data improves CNEP's performance. For example, when the data comes from a 2-modal distribution, the difference between CNEP-2 and CNEP-8 is negligible. As the number of modalities increases, the model must use more experts.

Table 4.2. MSE over generated trajectories.

	Unimodal	**Bimodal**	**Multimodal (4 modes)**
CNEP-Uni	0.042 ± 0.042	0.185 ± 0.057	0.120 ± 0.079
CNEP-UnSpec	0.033 ± 0.031	0.069 ± 0.096	0.050 ± 0.077
CNEP-Rec	**0.031 ± 0.028**	0.113 ± 0.045	0.064 ± 0.069
CNEP-MoE	0.046 ± 0.049	0.061 ± 0.071	0.040 ± 0.061
CNEP	0.046 ± 0.049	**0.058 ± 0.071**	**0.028 ± 0.049**

Table 4.3. MSE over generated trajectories.

	2 Modes	**4 Modes**	**6 Modes**
CNEP-2	0.005 ± 0.087	0.045 ± 0.038	0.063 ± 0.167
CNEP-4	0.005 ± 0.090	0.042 ± 0.038	0.033 ± 0.148
CNEP-8	0.004 ± 0.048	0.042 ± 0.04	0.032 ± 0.168

4.4.4. Evaluations on Simulation with High-Dimensional Trajectories

Real-world tasks that modern robotics research targets contain more informative representations thanks to advanced sensor technology and more realistic robotic actuators. Today, many static manipulators have 6 or 7 DoFs and humanoid robots have

tens of DoFs. Creating robotic controllers that take in high-dimensional sensory data as input to infer the state of the environment and produce high-dimensional control signals is a requirement for state-of-the-art LfD frameworks.

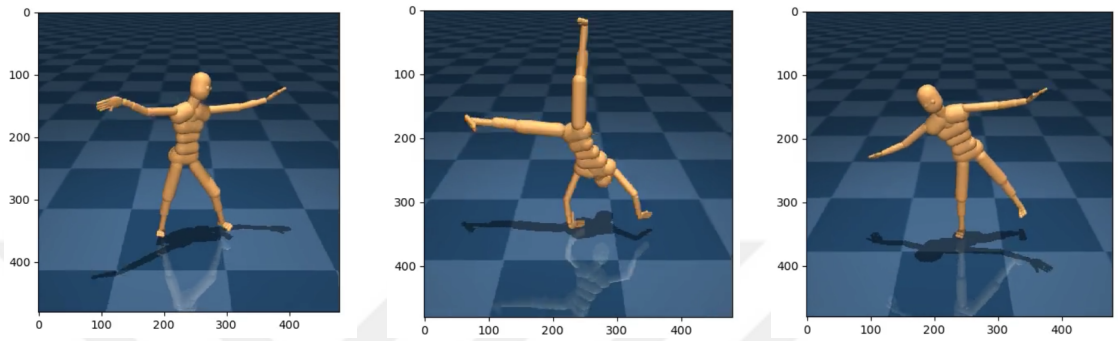


Figure 4.8. Using two 56-dimensional demonstration trajectories from [127], the CNEP model successfully recreated the cartwheel motion on a simulated humanoid.

To show the competence of CNEP against such high-dimensional data, a set of experiments are conducted. The expert demonstrations are taken from the CMU Mocap dataset [127], where actual stunts realize several motion behaviors, which are recorded using the motion-capture technology and producing 56-dimensional joint position trajectories.

First, a CNEP with only one expert module is trained with only two trajectories of a cartwheel motion. The aim here is to replicate the real-world demonstration data on the simulated humanoid model without actually controlling the agent. After training the model, it is asked to recreate the same motion, and the resulting trajectory is executed in the simulation. Snapshots given in Figure 4.8 show frames from the simulation.

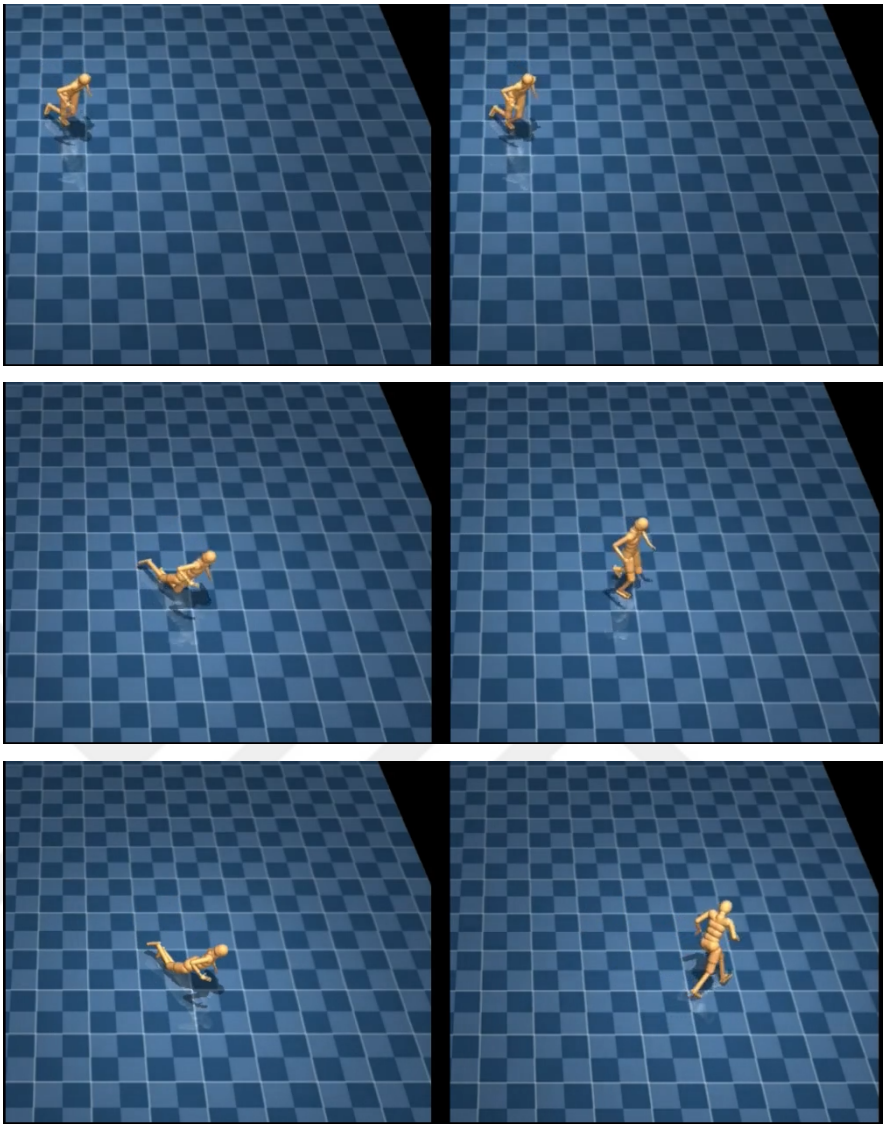


Figure 4.9. Using online sensor data of 287 dimensions, a CNMP (on the left) and a CNEP (on the right) are trained with walking and running primitives. CNEP performed better with smaller decoders, thanks to the specialization of the experts.

Upon successful results from the abovementioned experiment, we aim to show how a CNEP can be used to create an online robotic controller. Even though the data from the CMU Mocap dataset contains trajectories recorded by actual humans, it does not contain robotic sensory data or joint control actions that are needed to create the control policy on a humanoid. Therefore, the GPT agent, which is already trained on the CMU Mocap data as explained in [128], is used to generate 56-dimensional control signals for each joint on the simulated humanoid. While realizing the actions,

we recorded the sensory data of the humanoid as well, which contains 287-dimensional information about the proprioceptive state of the robot and the environment.

After generating the demonstration data, a CNEP with two expert modules is tasked to learn and differentiate between the trotting and wandering behaviors of a simulated humanoid robot. These two similar behaviors can be considered as different movement primitives for a social robot, and the distinction between them becomes important from the sociability perspective.

To show how distinguishing different skills may contribute to the learning procedure, we also trained a CNMP of the same size to conduct a fair comparison. Snapshots from the simulation are given in Figure 4.9. The result of this comparison shows that by making a distinction between two different skills, CNEP outperforms CNMP in resynthesizing the trained motions.

4.4.5. Learning from Real Robot Demonstrations

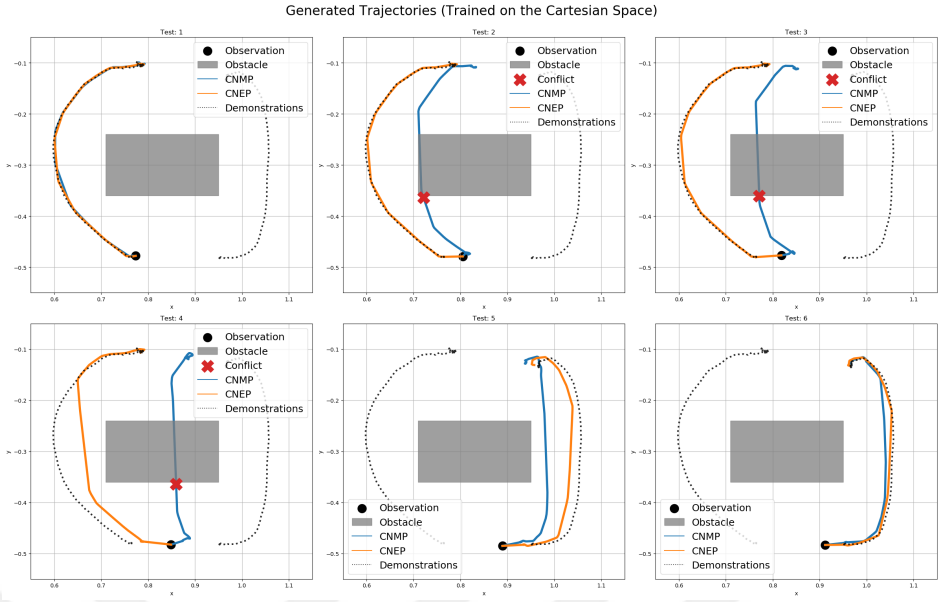
In this section, we evaluate the performance of our system on two real-world tasks using a robotic manipulator, the Baxter robotic platform [129]. In both experiments, demonstrations were collected following the kinesthetic teaching approach [7]. The first experiment explained in Section 4.4.5.1 demonstrates the learning and generalization capabilities of the CNEP model, where the robot is expected to realize an obstacle avoidance task. The second experiment illustrates how CNEP performs against high-dimensional sensory data on a more complex task that requires picking and placing objects in different configurations.

4.4.5.1. Obstacle Avoidance . The first experiment investigates the advantages provided by the CNEP model over the CNMP model in a multimodal obstacle avoidance task where two demonstrations that avoid obstacles from different sides were shown by an expert. The SM demonstrations include (1) the timestamped points of seven joints of the manipulator in the joint space and (2) the 7-dimensional pose of the end effector

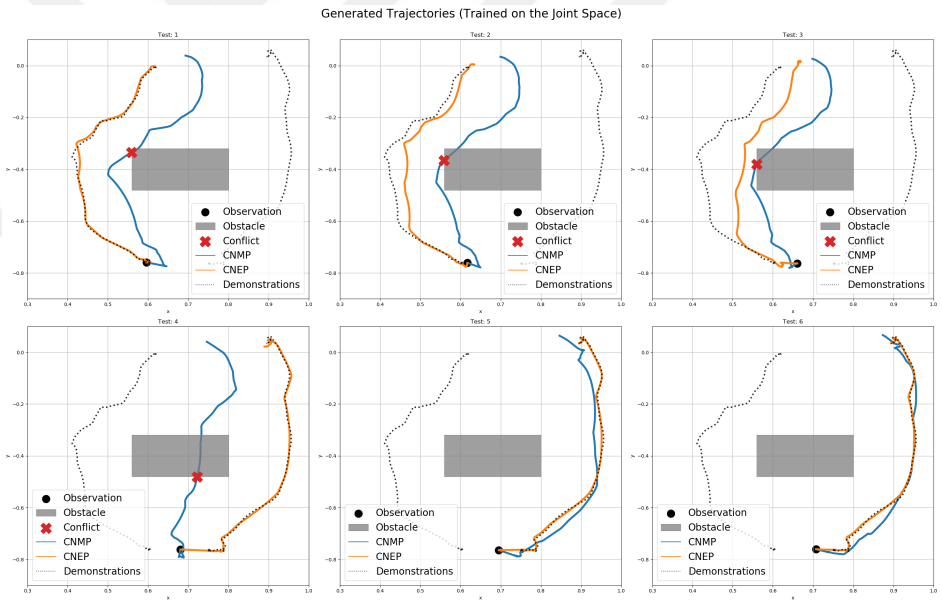
in the Cartesian space (Figure 4.10). We carried out evaluations both in Cartesian and joint spaces. The dataset from these demonstrations comprises two SM trajectories that move the robot arm from a start to an end position while avoiding an obstacle and grasping the target object by pressing a button.

Figure 4.10. Demonstrations of the obstacle avoidance skill are being performed by an expert. Kinesthetic teaching is used to generate sensorimotor demonstrations. Later, this data is used to train CNEP and CNMP models.

After training, both models were requested to generate the obstacle avoidance skill. To demonstrate the capabilities of the CNEP model, we chose to condition both the CNMP and CNEP on distinctive starting points. These conditioning points were selected to lie at the midpoint between the initiation points of the two demonstrations to compare the generalization capabilities offered by the models. In the first case shown in Figure 4.11, where models were trained with the trajectories of end-effector positions, generated values were passed through a PID controller (See Section 4.2.3.3) and an inverse kinematics module before the execution on the robot. Similarly, in the second case, which is shown at the bottom row of Figure 4.11, generated joint angle values are passed through a PID controller and a forward kinematics module in the second case for illustration purposes. Results in Figure 4.11 indicate a distinctive pattern, supporting our initial claim with trivial trajectories given in Section 4.4.1. This behavior is due to the mode-collapse issue, explained in [94, 130].



(a)

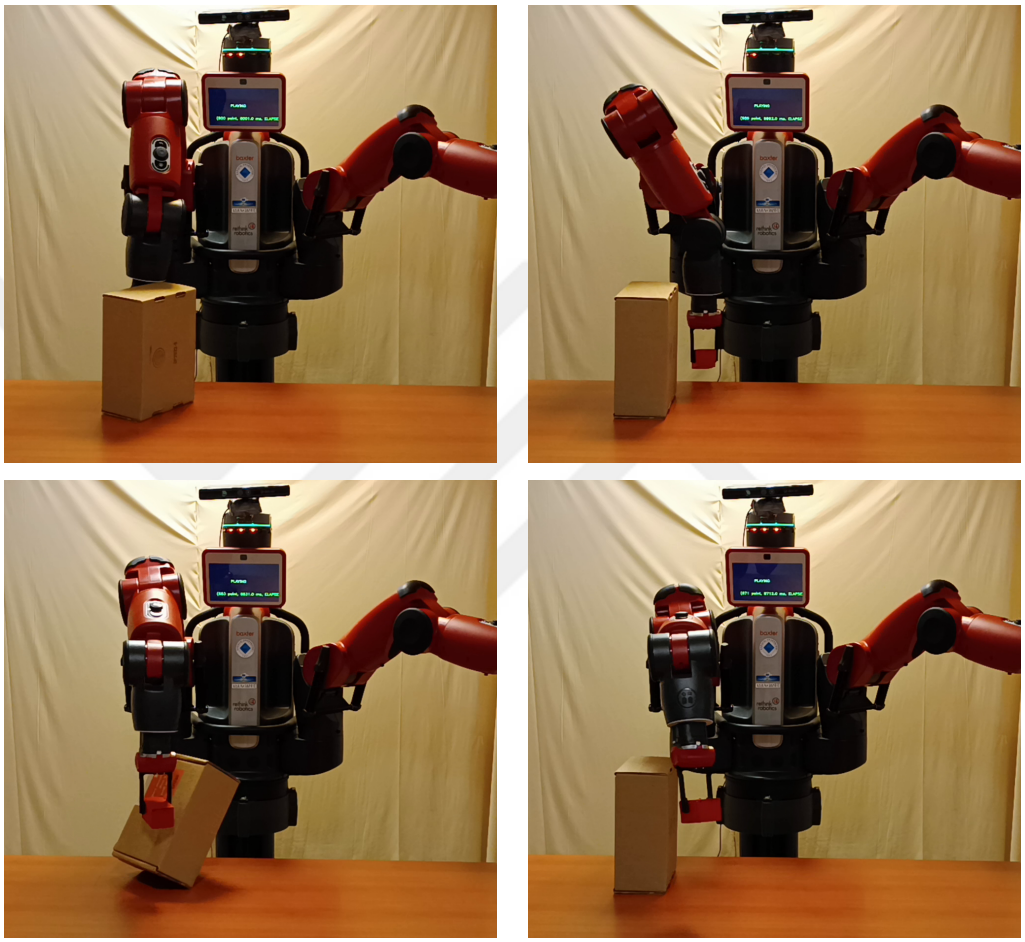


(b)

Figure 4.11. Comparison of trajectories generated by models in (a) Cartesian, (b) joint spaces. Using the same expert demonstrations (shown with dashed lines), CNMP-generated trajectories lead to collisions due to the mode collapse issue, while CNEP trajectories successfully avoid the obstacle. Refer to the text for details.

For this comparison, as shown in Figure 4.11, two CNEPs and two CNMPs are trained; one in the Cartesian space, shown in (a), and another in the joint space, shown in (b). Conditioned on an observation (shown with •), CNEP (shown in orange) chooses

one of the modes and generates motion trajectories closer to the expert, while CNMP (shown in blue) interpolates between modes, leading to collisions (shown with a red X) with the obstacle (shown in gray).



(a) (b)

Figure 4.12. In the left column, shown with (a), the generated trajectories by the CNMP model collide with the box in the middle. In the right column, shown with (b), CNEP only interpolates within demonstrations from the same mode and, therefore, only follows one of the possible modes. Eventually, CNEP successfully generates paths that avoid obstacles.

While the CNMP tends to interpolate between the provided demonstrations and, therefore, experiences a mode collapse, the CNEP demonstrates a preference for adhering closely to the demonstrated behaviors. To explain the implications of this behavior

for real-world robot tasks, we execute the trained models on the real robot and present the comparative results in Figure 4.12. Given these demonstrations, CNMP-generated trajectories lead to collisions, whereas CNEP-generated trajectories can safely avoid obstacles.

4.4.5.2. Grasping and Placing in High-Dimensional Spaces . The setup shown in Figure 4.13 is used for this set of evaluations. In these experiments, the robot is expected to pick wine glasses from the tabletop and place them on a designated dish rack. Each glass can be grasped from 2 different locations: (1) from the rim and (2) from the stem. If the robot grasps the wine glass from the rim, it can hang the glass to the side of the dish rack (Places 1 and 2 in Figure 4.13). If it grasps from the stem, it can put the glass upside down on the top of the dish rack (Places 3 and 4).

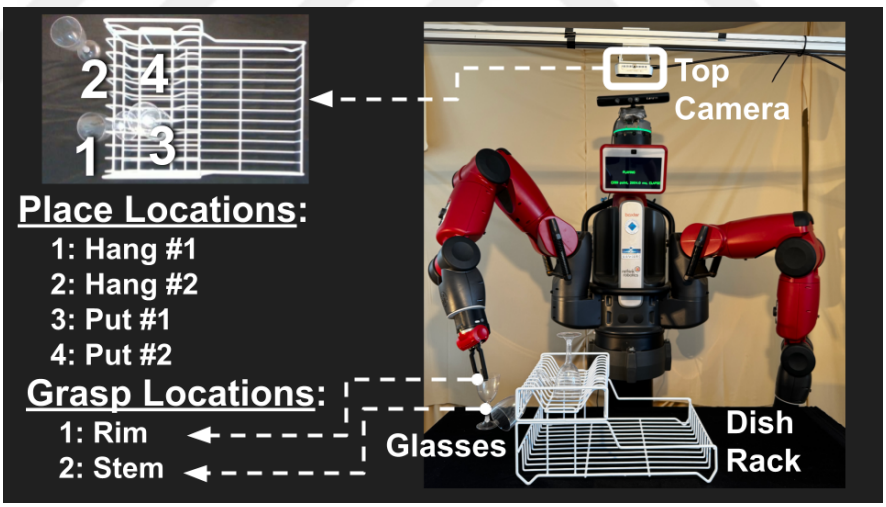


Figure 4.13. The top camera takes RGB pictures of the tabletop, which are used to extract spatial features of the objects. A glass offers 2 grasping locations for Baxter: (1) from the rim and (2) from the stem. Also, there are 4 placement options for each wine glass. Refer to the text for the details about the scenario.

Initially, the robot captures RGB images of the tabletop that has a resolution of 640x480 pixels. These images are then processed by the pre-trained object detection network MobileNet-v2 [131]. In this process, we exclude the classification layer at

the end of the MobileNet-v2 model to focus on extracting only the spatial features of the objects in the image. This extraction results in a 1280-dimensional feature array that represents the spatial characteristics of the detected objects. Next, trajectories of target skills are demonstrated by an expert. This procedure is displayed in Figure 4.14.

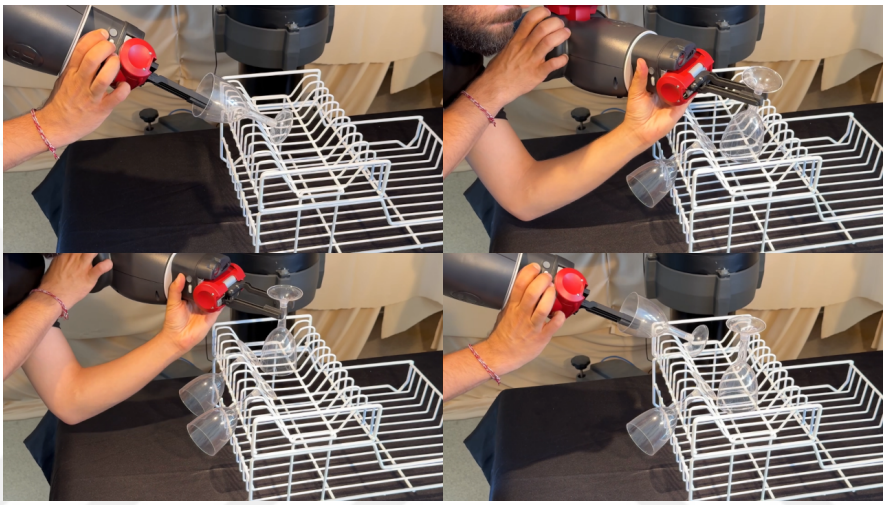


Figure 4.14. Using kinesthetic teaching, 40 different pick-and-place demonstrations are realized by an expert in 4 different scenarios. Recorded trajectories are used in the experiments given in this section.

Extracted feature arrays are combined with demonstration trajectories to form the input for CNEP. The complete input for the CNEP model is a list of 1288-dimensional trajectories: the 1280-dimensional feature array from MobileNet-v2, the 7-dimensional pose of the end-effector, and the status of the gripper (whether it is open or closed). Thus, the system integrates computer vision through MobileNet-v2 with the demonstration trajectories to control the robot executing this complex manipulation task. In the following experiments, the trained CNEP is used.

(i) Testing individual skills: After training CNEP, we used the same input shown in Figure 4.14 to train a ProMP and a GMM to compare the trajectory generation performances. When the data contained 1288 dimensions, both approaches failed

to model trajectories. Therefore, the dimensionality of the data is reduced following the technique given in [132]. In the testing phase, ProMP and GMM models failed to grasp the glass, while CNEP successfully grasped, picked, turned it upside down, and placed it, as shown in the demonstrations. Figure 4.15 presents pictures from this comparative test.

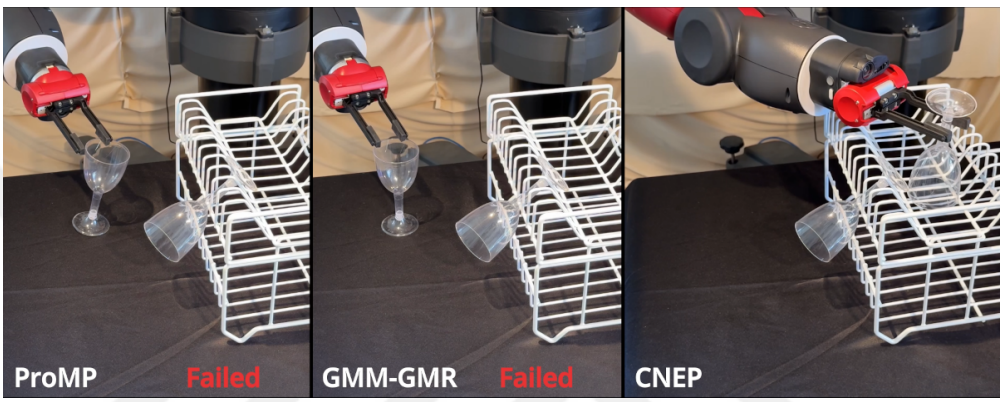


Figure 4.15. 40 pick-and-place demonstrations are provided. 3 models were trained on this data: a ProMP, a GMM, and a CNEP. Then, the glass was placed at a new location, and the models were conditioned on this observation. ProMP and GMM failed to grasp the glass properly, CNEP successfully carried out the entire task.

(ii) Testing the online conditioning and on-the-fly adaptation: Conditioned on the real-time video feed from the top camera, the position of the end-effector, and the status of the gripper, the CNEP model can act as an online controller, producing real-time control commands to realize target skills. This claim is supported by another set of experiments, which is displayed in Figure 4.16. To assess our hypothesis, using the same setup we introduced in the previous experiment, we tasked the same CNEP to pick up a glass from the table and put it on the dish rack by either hanging or placing it upside down. As CNEP was creating motion commands to realize the most appropriate behavior among the demonstrated skills, we changed the tabletop configuration twice during the execution.

Starting with the configuration shown in Figure 4.16(a), CNEP decides to use the skill learned by the expert-0 to grasp the glass from the stem and place it

on the dish rack because when only Hang#1 is occupied, the taught skill is to grasp the glass from its stem and to place it upside down on the location Put#1 in Figure 4.13.

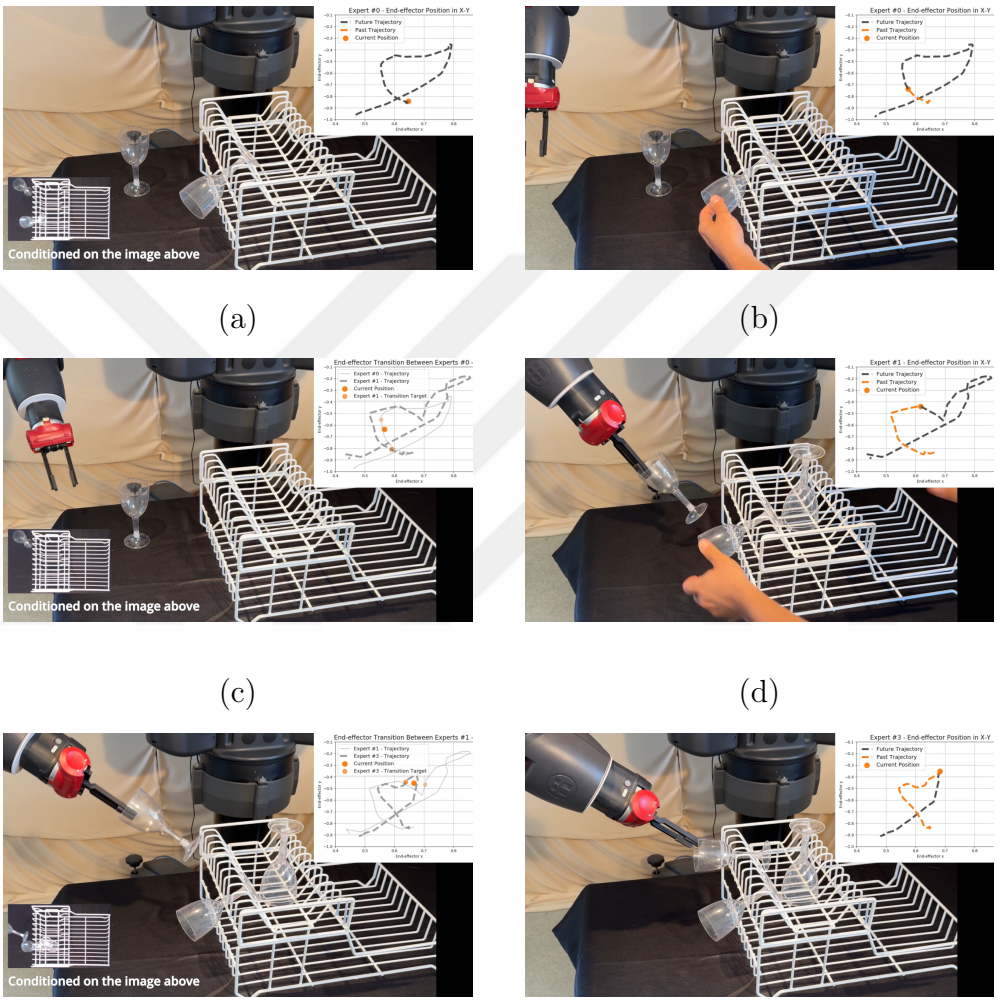


Figure 4.16. When the tabletop configuration changes during the execution of the trajectory, CNEP can adapt by continuously conditioning on the current sensory data and switching among experts on the fly. Please refer to the text for details.

During execution, while CNEP was moving towards a stem grasp, the hanged glass was removed, and the Hang#1 location was freed, as shown in Figure 4.16(b). Correspondingly, CNEP decides to transition from expert-0 to expert-1

to grasp the glass from the rim and hang it on the side of the dish rack, as shown in Figure 4.16(c), since CNEP is taught to hang the glass on Hang#1 in such cases. As it grasped the glass from the rim and was going towards the Hang#1 location, the tabletop is changed again, as given in Figure 4.16(d). This time, we occupied Hang#1 and Put#1. CNEP accordingly switched its behavior and the responsible expert from expert-1 to expert-3 in Figure 4.16(e), which completed the task by hanging the glass to the Hang#2 location, the available spot on the dish rack, as shown in Figure 4.16(f). In conclusion, our system successfully reacted to the configuration changes by changing the responsible expert and, hence, the produced control commands, as shown in Figure 4.16.

4.5. Conclusion

In this study, we introduced an LfD method, namely the Conditional Neural Expert Processes. CNEP is proposed to improve the modeling and generation capabilities of LfD systems when available demonstrations correspond to diverse, multimodal sensorimotor trajectories. This is achieved by the utilization of (1) the novel architectural components, the Gate Network, and the experts, and (2) the novel components of the loss function, the batch entropy, and the individual entropy.

Our experiments demonstrated that CNEP is a robust LfD approach for modeling and generating robotic skills even in real time. It successfully models intersecting multimodal or significantly different trajectories. It has better performance than baseline methods and effectiveness in real robot experiments. The number of experts is set manually and can be optimized as a hyper-parameter in the future. One limitation of CNEP is that, as a probabilistic framework, it does not guarantee passing through the observation points and requires using a higher-level module, such as a PID controller, to guarantee precision at observation points. Additionally, similar to other neural network-based approaches, CNEP cannot extrapolate outside the training range.

5. MODELING PERIODIC SKILLS WITH POSITION ENHANCED MOVEMENT PRIMITIVES

5.1. Introduction

Learning from Demonstration (LfD) is a paradigm where a robot learns a policy by observing samples provided by an expert. LfD has recently gained attention for enabling robots to learn intricate policies without the need for explicit reward definitions. Compared to Reinforcement Learning (RL), LfD accelerates the policy-learning process by leveraging expert-provided demonstrations as an initial guide, reducing the exploration burden. Moreover, LfD bypasses the need for reward engineering, which can be challenging in tasks with complex or poorly specified reward structures, eliminating the risk of misleading policy optimization [133].

Applying LfD to teach a robot new skills necessitates a mathematical representation of the demonstrated sensorimotor (SM) trajectories. Dynamic Movement Primitives (DMPs) [11] are one of the prominent methods, providing a mathematical framework to encode demonstrated SM trajectories in terms of movement primitives (MPs). The DMP formulation represents MPs as stable nonlinear dynamical systems [134]. Despite its widespread use, the original DMP formulation is designed to encode a single SM trajectory, restricting its ability to represent a broader range of the intended policy. This constraint reduces the generalizability of learned policies to complex and dynamic real-world states, posing challenges for tasks requiring diverse or adaptive behaviors.

Many tasks in the world are solved by the rhythmic application of the same motor primitive, such as hammering a nail or opening a bottle cap. For such skills, considering and leveraging the periodicity in the demonstrations enables learning descriptive representations. Periodic DMP (PDMP) has been proposed in [45] to leverage the cyclic traits of such tasks to enhance the learning of periodic MPs. However, they

suffer from the same limitations as the original DMPs; they can only model a single trajectory, not a distribution of trajectories.

When one or more experts are asked to provide multiple SM demonstrations of the same skill, their personal preferences or abilities, such as speed or mobility, influence the synchrony of the SM trajectories in the demonstration set. For periodic trajectories, the differences in the demonstrator's behaviors may correspond to different frequencies or phase shifts. This phenomenon causes challenges for approaches that rely on a linear and monotonically increasing phase value and do not explicitly leverage the periodicity of the demonstration. A solution is to apply pre-processing to align them [135]. While this approach offers a solution, it can be a demanding procedure and can render some of the valuable demonstrations unusable.

Advances in statistical techniques have significantly influenced DMPs and other LfD approaches, offering an improved representational power to encode trajectory groups as MPs, enabling them to learn more complex policies. Probabilistic Movement Primitives (ProMPs) are proposed in [12] to model distributions of trajectories. The ProMP model is shown to work well against multiple trajectories of multiple degrees of freedom (DoF). In [136], the modeling performance of ProMPs is shown to outperform many other approaches when the number of kernels is increased. On the other hand, it is shown in [18] that ProMPs face challenges when the demonstration set is composed of multimodal distributions of the same skill, which is the case in this study.

An even increased representational power is offered by deep-learning (DL) based LfD frameworks, such as Conditional Neural Movement Primitives (CNMPs) [16]. CNMPs encode MP-based policies as neural networks, capturing complex nonlinear relationships inside the demonstrations. Unfortunately, their utilization in rhythmic tasks is limited as they do not leverage the periodicity of the task at hand but rely on a linear and monotonic phase signal.

In this study, we propose a novel encoder-decoder architecture called Position-Enhanced Movement Primitives (PEMPs) to enable DL-based encoding of MPs for periodic skills. PEMPs offer increased representational modeling against multiple demonstrations of the same skill with different frequencies and phase shifts by borrowing ideas from the positional encoding technique introduced in [47]. The utilization of trigonometric positional encoding instead of the linear phase provides an oscillator that biases the underlying neural network to discover rhythmic patterns in the demonstration data. We first present evaluations of the PEMP model on synthetically generated trajectories to explain the proposed model better. Later, we present comparative assessments with a simulated robotic hand in a nail-driving task. Finally, to show the applicability in real-world tasks, we provide our evaluations of the PEMP model with a real UR10 manipulator on a bottle-opening task. The results of these experiments support our hypothesis that positional encoding indeed improves the representational power in encoding periodic SM trajectories in terms of MPs.

5.2. Method

5.2.1. Background

The key observation behind the PEMP model comes from the Fourier series, which states that any continuous periodic signal can be expressed as a combination of sine and cosine functions. To provide sine and cosine waves, we use the technique of positional encoding, whose original formulation is introduced in [47], that uses the sine function for even positions and the cosine function for odd positions in the input array, computed as

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right), \\ PE_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right). \end{aligned} \tag{5.1}$$

Even though the approach in [45] does not leverage deep learning techniques to model distributions of demonstration trajectories, the research provides valuable insights into

modeling periodic tasks. PDMPs use nonlinear oscillators to provide the phase signal. The PEMP model, on the other hand, does not use nonlinear oscillators. Positional encoding (PE) given in Equation(5.1) produces a linear oscillator, which introduces an inductive bias that favors capturing periodic patterns. The nonlinearity required to model the periodic trajectories comes from the nonlinear activation of the neural network itself.

5.2.2. Proposed Method: Position-Enhanced Movement Primitives

PEMP is an LfD framework that is specifically tailored to capture rhythmic patterns in the observed demonstrations and encode periodic skills in terms of MPs. It is able to encode and synthesize motions in both joint and Cartesian spaces. To generate SM trajectories following the trained characteristics, the model must be conditioned on a specific observation point. This can be any point, such as the starting point, the endpoint of the desired SM trajectory, or a set of intermediary via points. Passing through the conditioning points is not guaranteed because the underlying neural network utilizes Gaussian distributions to model trajectories, however, the PEMP model can be coupled with a closed-loop feedback controller, such as the PID controller, to improve precision.

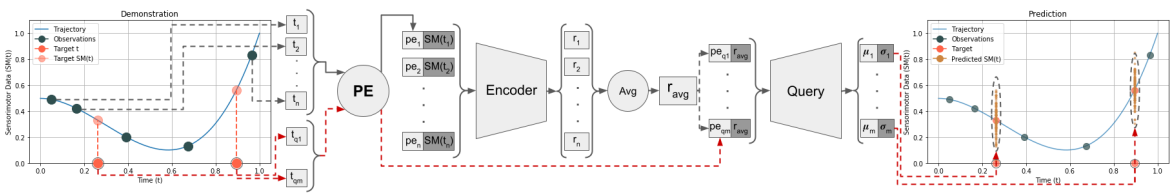


Figure 5.1. The proposed PEMP model comprises two neural networks, an Encoder and a Query. The operation shown as PE is the implementation of Equation(5.1), which outputs a multidimensional oscillating phase. The encoded representations are aggregated to obtain r_{avg}, which is concatenated with query positions to produce the responses of the system. For more details, please refer to the text.

Given in Figure 5.1, the training procedure starts with a random sampling of a set of observation points from the demonstration trajectory. Remembering that the definition of a trajectory is an ordered collection of timestamped points, each sampled observation has a timestamp normalized in the unit interval for the sake of consistency. This value is passed through the PE function given in Equation(5.1) to bring the abovementioned oscillatory characteristic to the input of the encoder. The encoder network receives its input as concatenations of PE values with corresponding SM values. If desired, any task-specific parameter can also be appended to the input at this stage. The handling of the task-specific parameters is similar to that in CNMPs.

The encoded representations are aggregated via an averaging operation to produce r_{avg}, a compact representation of the entire SM trajectory in the latent space. In turn, r_{avg} can be concatenated with any query point and passed through the query network to produce the predictions of the system as Gaussian distributions at query points. Given $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as the output of the PEMP model, the training loss of the system is computed as a negative log-likelihood error

$$\mathcal{L} = -\, log\, P(\, SM(t_q) \mid \boldsymbol{\mu}_q,\, softmax(\boldsymbol{\Sigma}_q)\,), \tag{5.2}$$

which is backpropagated through both networks to adjust their weights. Upon training with the loss given in Equation(5.2), the system can be queried for any $(t_q \mid SM(t_q))$ pair, and the system can generate full SM trajectories.

5.3. Experiments and Results

To first depict the idea and then show its applicability to real-world tasks, we present the evaluations of the PEMP model and comparative assessment against the benchmarking approaches —ProMP and CNMP —in order of complexity, increasingly approaching the real world. Therefore, to provide an insight into PEMP and its benefits, the assessments start with artificially generated SM trajectories of growing complexities. Subsequently, we provide a comparison on a nail-driving task with a simulated robotic hand requiring a 26-dimensional control signal. Finally, we present a comparison of all approaches on a bottle-opening task with a real manipulator, UR10.

In all experiments, a CNMP, a PEMP, and multiple ProMP instances are trained. Since the ProMP model is sensitive to the number of underlying basis functions as explained in [136], we trained 30 different ProMPs with different numbers of basis functions ranging between 5 and 50. We present the results of the best-performing ProMP. Also, we noticed that inputting the frequency of the trajectory improved the performance of the models by giving them the opportunity to estimate the angular phase of the cyclic pattern. For the CNMP and PEMP instances, the frequency information is input as a task parameter concatenated to the input pairs. In the case of ProMPs, we followed the approach explained in [137], which combines a Gaussian Mixture Model (GMM) with each ProMP to provide the context. All of the comparative results presented below are found to be statistically significant, and the multimedia showing these results is given in the repository shared on the first page.

5.3.1. Comparison on Synthetic Sensorimotor Trajectories

The evaluations provided in this section aim to underline the proof of concept shaping this study on synthetically generated data. Assuming the trajectories shown below correspond to SM trajectories that are rhythmically applied by some experts, these tests demonstrate that the PEMP model utilizes the periodicity inside the control signals with different frequencies and phase shifts better than models relying on a linear and monotonically increasing phase value.

5.3.1.1. Simple Demonstrations. We gathered a dataset of 200 sinusoidal trajectories with the abovementioned characteristics. The initial phase and the frequency of each trajectory are sampled randomly. Subsets of the dataset are illustrated in Figure 5.2. Using the entire dataset, we trained a CNMP, a PEMP, and 30 different ProMPs, each with a different number of basis functions. An example test with novel conditioning points is given in Figure 5.2 for each frequency value.

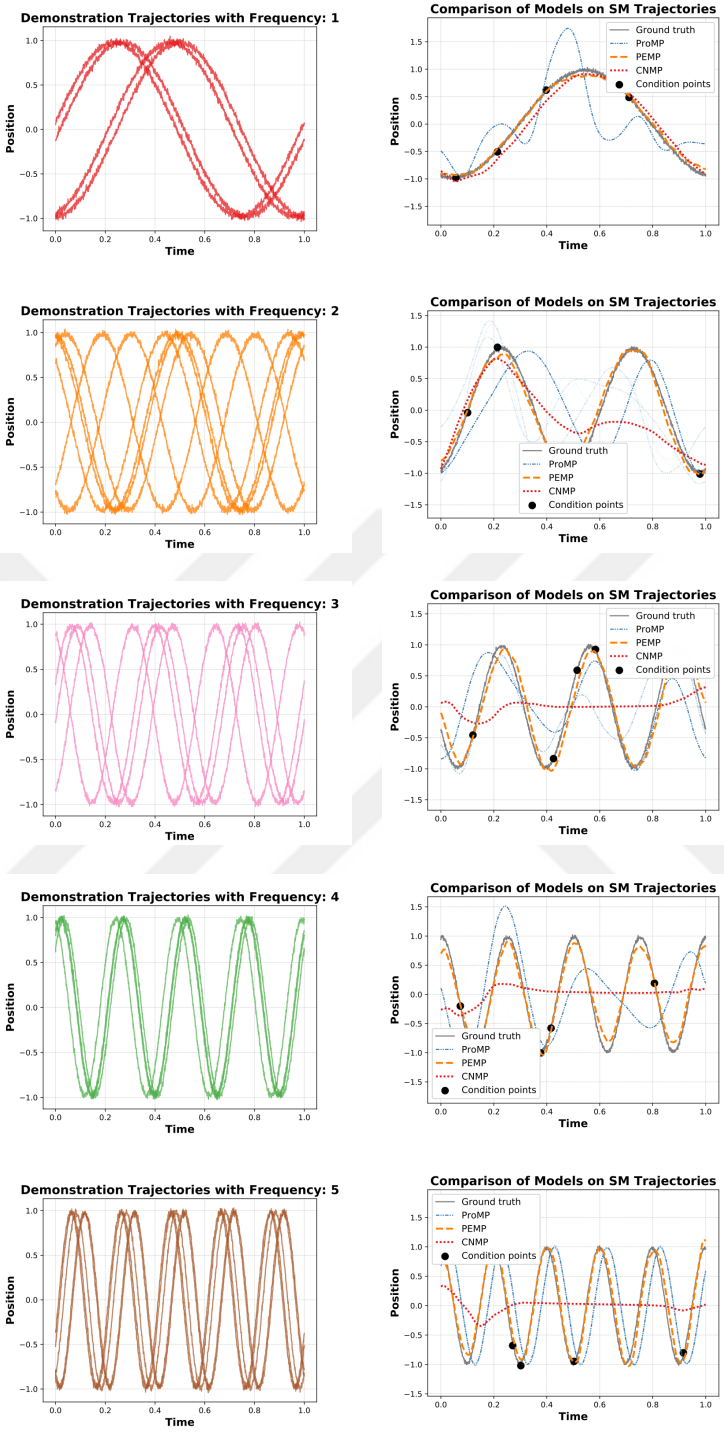


Figure 5.2. Multiple ProMPs, a CNMP, and PEMP instances are trained using the entire dataset, whose subsets of varying frequencies are plotted on the left-hand side. After training, each model is asked to construct trajectories conditioned on randomly sampled novel points. On the right-hand side, examples of each test case are presented. Overall results are given in Table 5.1.

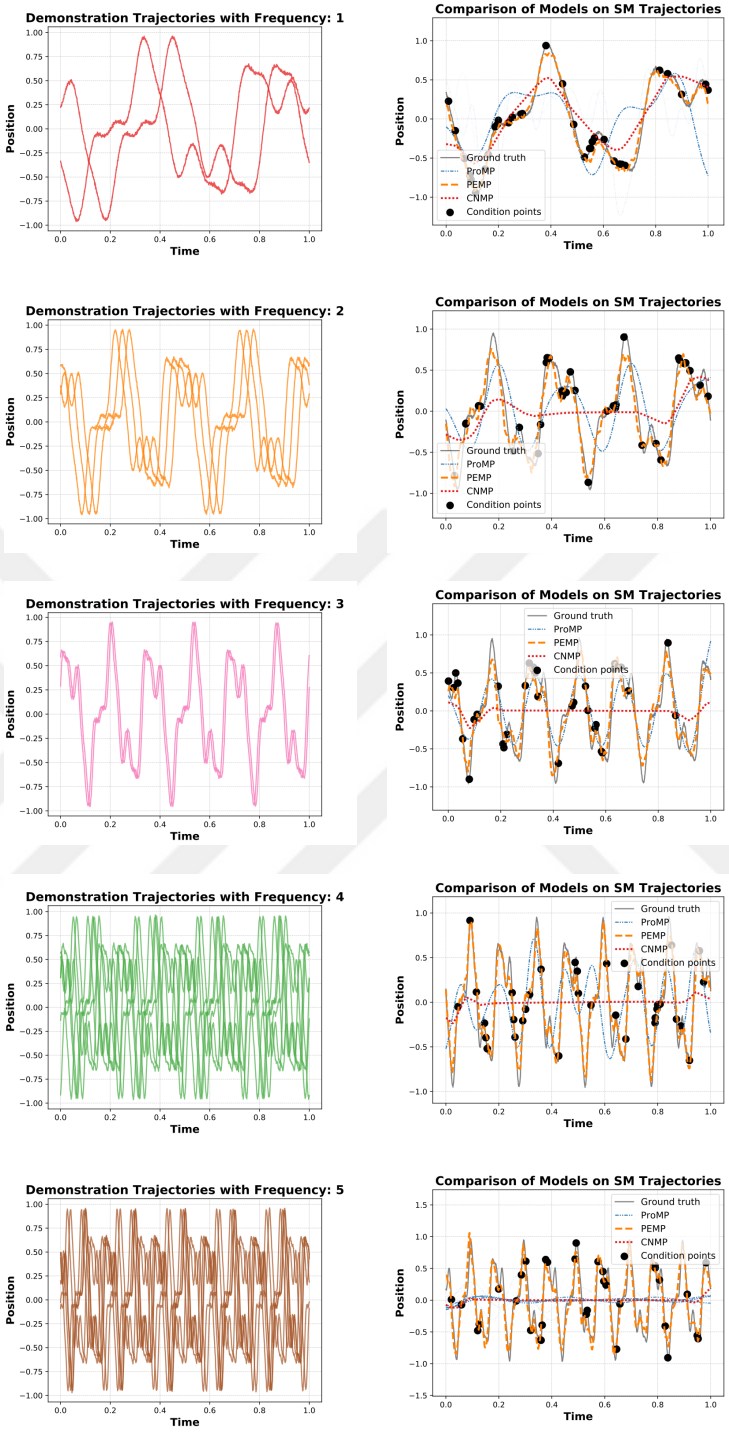


Figure 5.3. On the left, subsets with varying frequencies of the entire demonstration dataset. Multiple ProMPs, a CNMP, and a PEMP instance are trained using the dataset. After training, each model is asked to construct trajectories conditioned on randomly sampled novel points. Examples of synthesized trajectories are given on the right-hand side. Overall results are gathered in Table 5.1.

5.3.1.2. Complex Demonstrations. After successful evaluations with simplistic trajectories, we tested ProMP, CNMP, and PEMP models on more complex data. The motivation behind this set of tests is to evaluate the performances on trajectories that are much more similar to real-world control signals.

While keeping the controlled settings, we created a dataset of complex trajectories, a linear combination of different trigonometric functions. Figure 5.3 illustrates subsets of this dataset. Additionally, on the right-hand side of the same figure, we provide trajectories generated by these models using randomly sampled novel conditioning points.

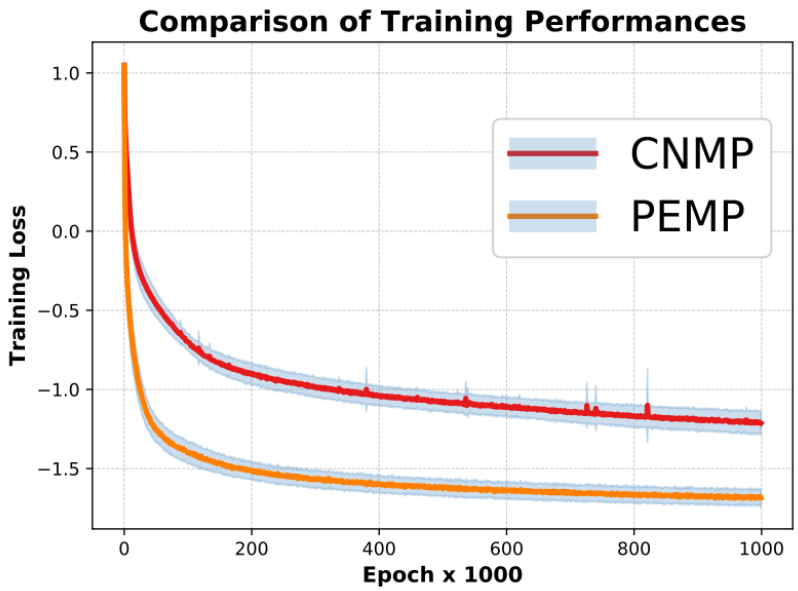


Figure 5.4. Comparison of training losses shows that the proposed model, PEMP, makes smaller errors and converges faster even with considerably smaller model sizes on periodic data. Refer to the text for details.

5.3.1.3. Analysis. The hypothesis of this study was that the integration of positional encoding biases the underlying function estimator to capture periodic structures inside demonstration trajectories. Evaluations with the synthetic data provided controlled environments to test this hypothesis. We noticed that the reconstruction losses attained by the PEMP model during training were constantly better than those by CNMP. Also,

the convergence is achieved much faster. To illustrate this observation, we compared the training losses generated by both models in Figure 5.4 over multiple training sessions on the simple dataset. Moreover, Table 5.1 presents the MSE computed between the ground truth trajectory and reconstructed trajectories over 1000 tests.

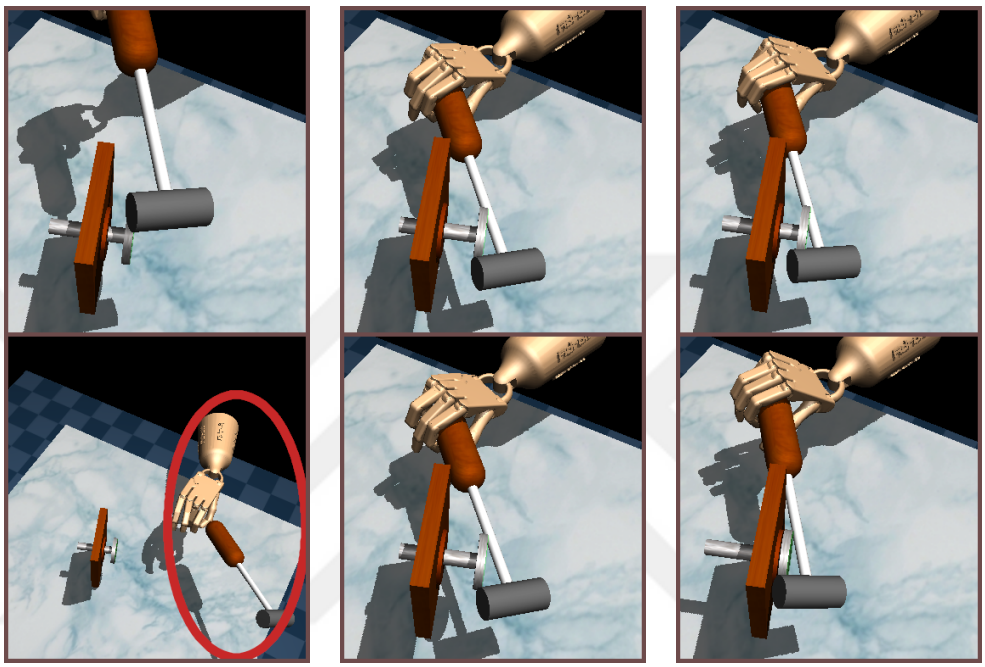
Table 5.1. Comparison of Mean-Squared Errors of ProMP, CNMP, and PEMP.

	Simple	**Complex**
ProMP	0.015 ± 0.005	0.010 ± 0.003
CNMP	0.015 ± 0.007	0.013 ± 0.001
PEMP	$\mathbf{0.007 \pm 0.009}$	$\mathbf{0.006 \pm 0.005}$

5.3.2. Test on Simulation

Demonstration trajectories on synthetic datasets are composed of 1-dimensional functions. After the successful applications under the controlled settings, we proceeded to assess the PEMP model on more difficult and realistic SM trajectories. In this test, we used the Adroit Hammer environment [138] from the Gymnasium interface [139]. This environment contains a robotic hand, requiring a 26-dimensional control signal as joint values. The task is to use the robotic hand to drive the nail into the board using a hammer. The positions of the objects in the scene are given as observations; therefore, no visual processing is applied. Using prerecorded expert demonstrations, we trained multiple Contextual ProMPs, a CNMP, and a PEMP. After training, all models were conditioned on the starting step of the environment, and their performances were evaluated. In Figure 5.5, example screenshots illustrating the results are provided. For this task, the maximum depth the nail could be driven was approximately 4 cm. The PEMP model in Figure 5.5(c) was able to drive the nail to the maximum depth in most cases, although it occasionally fell slightly short, demonstrating its ability to learn and execute the periodic hammer-swinging skill effectively. In contrast, the ProMP model delivered a single powerful strike, exerting relatively high force, but failed to maintain the grasp of the hammer afterward, as shown in Figure 5.5(a). The CNMP model in

Figure 5.5(b), while able to retain the grasp of the hammer, failed to execute rhythmic swinging, highlighting its inability to capture the periodicity of the task. Figure 5.6 presents a comparison of the performances of the three approaches on this task.



(a) ProMP (b) CNMP (c) PEMP

Figure 5.5. ProMP, CNMP, and PEMP methods were tested on a simulated nail-driving task. PEMP successfully captured the periodic hammer-swinging skill and consistently achieved the best performance across trials.

5.3.3. Application in the Real World

To assess the applicability of PEMP in the real world, we set up a bottle-opening task with a 7-DoF UR10 manipulator. A 3-finger gripper is attached to the end-effector of the manipulator for grasping the bottle cap. Figure 5.7 shows the environment configuration where the task is to grab the bottle cap in its initial configuration, shown on the left, and twist the cap open to take it off, as shown on the right.

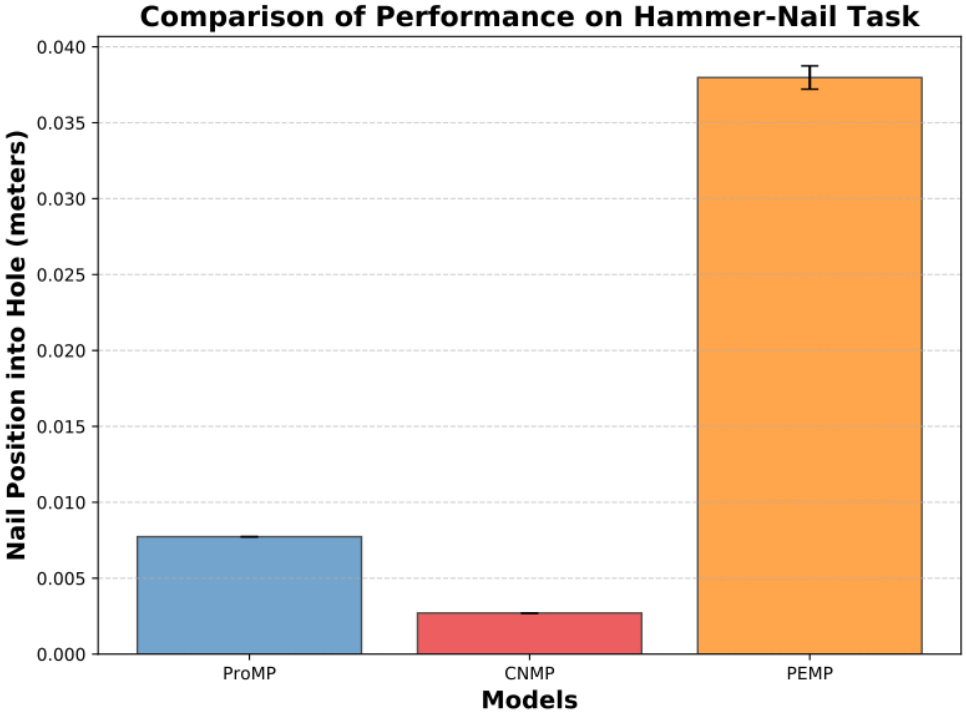


Figure 5.6. Even though the ProMP model performed better than CNMP, it failed to maintain the grasp after the first successful strike of the hammer. Whereas CNMP was able to maintain its grasp, it did not learn to deliver repeated strikes, although they were given in expert demonstrations. PEMP managed to learn the periodic movement primitive rather successfully to drive the nail until the end in most cases.

To collect demonstration trajectories of the task, we manually coded a control policy that randomly samples the frequency of the demonstration trajectory from an interval of 3-6. An important point about the task is that upon grasping the cap, only the third joint in the wrist (Wrist-3) applies a rotary force, and the other joints are kept still. This way, 20 SM trajectories of 8 dimensions are recorded —7-dimensional joints and 1-dimensional gripper values. Figure 5.8 illustrates Wrist-3 joint values in these recorded trajectories. Other dimensions are omitted to disintricate the figure.

After normalization, these trajectories are used to train one CNMP and one PEMP model simultaneously for fair comparison. Two example tests during training are presented in Figure 5.9. As with the cases presented above, the PEMP model achieved much lower loss values, indicating more successful modeling of the skill.

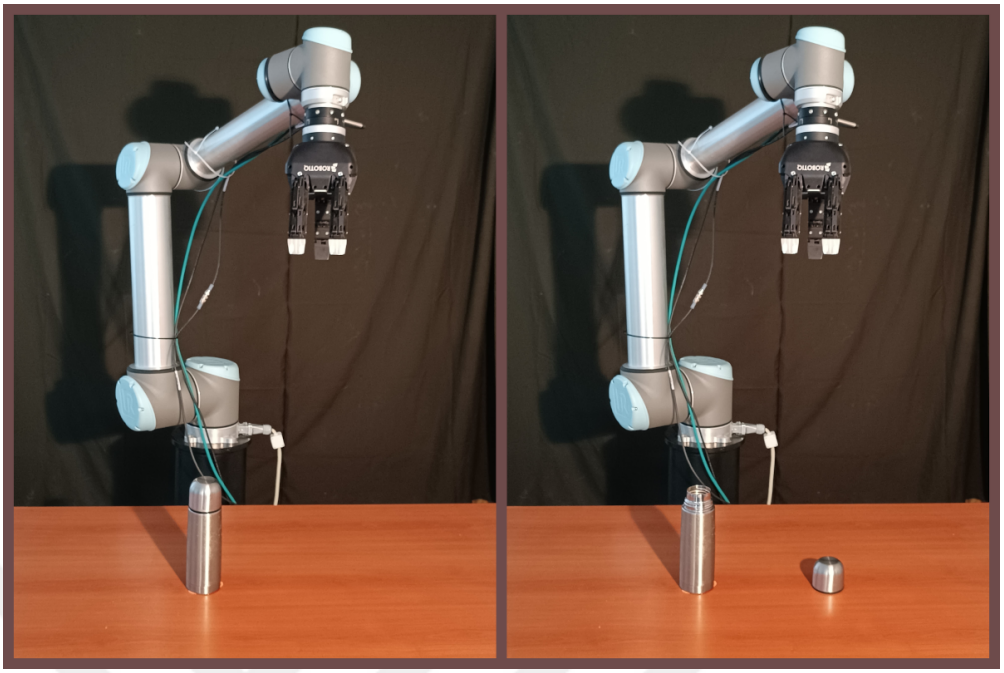


Figure 5.7. The configuration of the tabletop for the bottle-opening task. This task includes a UR10 robot and a thermos whose cap can be opened by rotating.

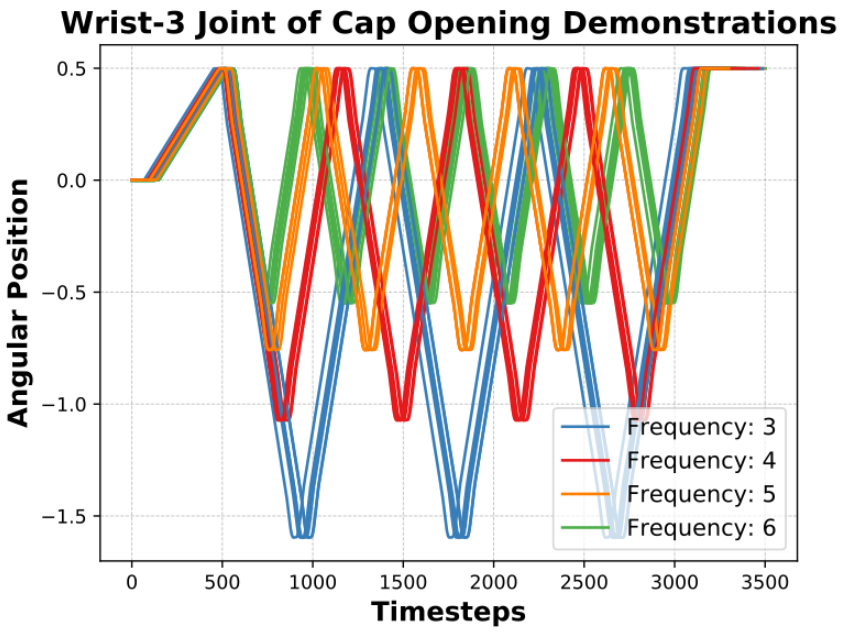


Figure 5.8. Position values of the Wrist-3 joint in all expert demonstrations. While other joints are kept still, applying these signals to the Wrist-3 joint leads to the successful execution of the target skill.

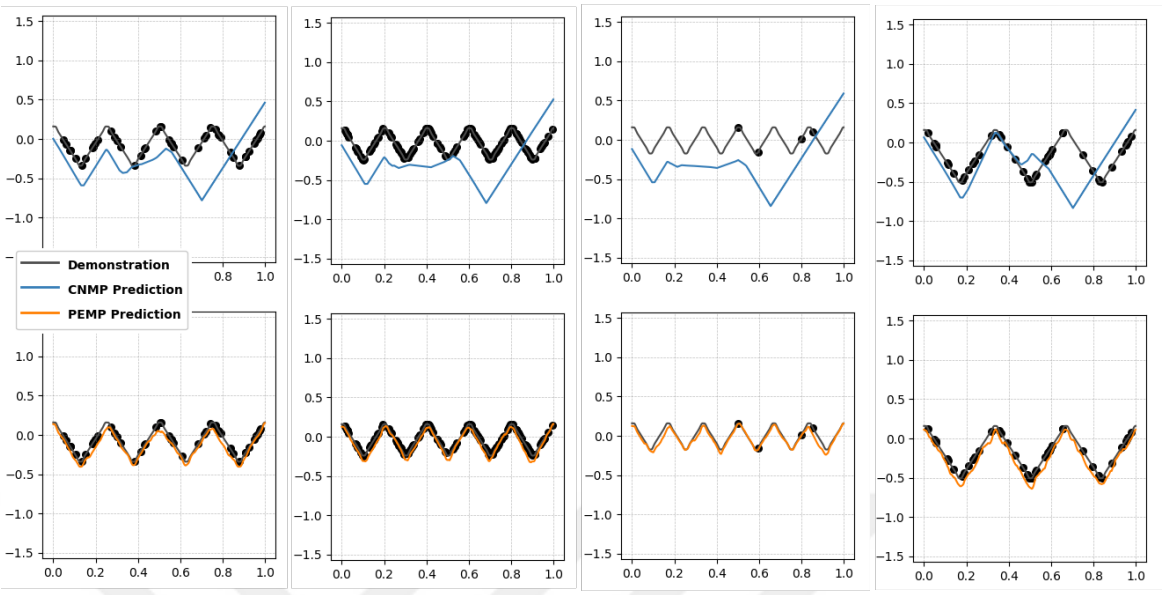


Figure 5.9. Comparison of modeling performances between CNMP and PEMP on UR10 data. PEMP is more successful in learning the rhythmic Wrist-3 joint data.

5.4. Conclusion

In this paper, we introduce the PEMP model, which is a DL-based LfD technique offering improved modeling performance for periodic skills. It utilizes sinusoidal positional encoding instead of the linear time-based phase variable to discover rhythmic patterns in the demonstration data. Compared to the alternative approaches, the PEMP model offers several advantages, including:

- faster convergence against DL-based techniques as it leverages the repetition in the SM trajectories.
- improved data utilization as it does not need trajectories with different characteristics to be aligned.
- improved performance in synthesized periodic motions.

One limitation of this approach is that it requires slight fine-tuning when the demonstration trajectories contain a combination of periodic and nonperiodic motions. For such cases, a combination of linear and sinusoidal phases can be input into the

system, which is left as a future work. Another limitation is that it requires all demonstration trajectories to be the same length. Considering periodic skills, this assumption might limit data utilization as some experts prefer solving tasks faster, leading to different trajectory lengths. Learnable positional encodings can be utilized to find better encodings that can explain the relative position of the SM data in variable-length trajectories.

6. DISCUSSION

This thesis aims to contribute to robot learning by proposing novel and practical models that can be used in modeling and regenerating an expert's navigation and manipulation behaviors, following the LfD paradigm. This motivation was based on the assumption that navigation and manipulation tasks comprise the majority of the practical uses of our expectations from a modern robot. It is a common practice in research to turn to nature for inspiration, and until it is no longer valid, the most intelligent being in nature is the human. Hence, the most practical way to come up with solutions to most of the problems we try to tackle in robotics is to imitate the human. Neuroscience research provides evidence of MPs in humans' sensorimotor behaviors. Therefore, we believe that MP formalism gives robotics research an invaluable direction to pursue to enhance the functionality of robots. While this belief leads to limiting assumptions, such as people knowing the best methods for the problems we address, accepting this as a fact is only practical.

6.1. Social Navigation for Mobile Robots

With this mindset, we first focused on enhancing the sociability of mobile robots by improving their navigation behaviors to align more closely with human norms. Compared to traditional navigation systems that prioritize collision avoidance and efficiency, the approach presented in Section 3 places a stronger emphasis on social compliance. Previous studies have primarily focused on physical safety, but this work extends the focus to comfort by adhering to proxemics and other social norms. The proposed data-driven navigation framework utilizes a couple of CNPs that effectively encode trajectories that capture human intentions, thus improving the robot's ability to navigate more socially. By learning from real-world human data, the developed framework provides a promising approach to adapt to various social contexts and respond appropriately to dynamic environments, exhibiting social competence in robotic navigation. Eventually, this study demonstrates that incorporating navigation MPs of humans into

the mobile robot navigation pipeline significantly reduces discomfort in human-robot interactions.

We must admit without reserve that this study focuses only on one aspect of human navigation behavior and does so only partially, aiming to model the spatial aspects of collision avoidance behavior. However, as discussed in [67], humans exhibit many more behaviors during navigation, such as leader-following or grouping. Moreover, after working closely on the subject, we believe that humans' navigation behavior is also shaped by temporal features and not only by spatial features. We present our ongoing study in [37]. Additionally, we believe that a holistic approach needs to consider the constraints imposed by the kinodynamic characteristics of the robot. One cannot expect a differential-drive robot to go laterally like people do when they politely give way to strangers. As a result, while incrementally effective, our efforts are still far from providing a complete solution to the social navigation problem.

6.2. Modeling MPs from Multimodal Demonstrations

Research, such as [35], has made it explicit that humans exhibit a multimodal behavior during navigation. The story behind CNEPs, explained in Section 4, started with this observation. However, multimodality is also a problem for manipulation tasks, substantially affecting the performance of MP-modeling frameworks due to the mode-collapse issue. CNEP aims to leverage the entropy in the latent space of the proposed encoder-decoder model to utilize smaller but effective experts to learn multimodal behaviors. The authors of [44] criticize the methods using either information projection or moment projection techniques for underrepresenting the multimodal solutions. CNEP provides a combined solution where moment projection is achieved by individual experts and information projection is achieved by using entropy as an arbitrator among experts. The CNEP model is the apple of this thesis, but its performance in modeling navigational behaviors is yet to be assessed. Furthermore, it currently uses a fixed number of experts, which should be dynamically adjustable for better coverage of the latent space.

6.3. Modeling Periodic Skills

During the assessments of the CNEP model, we noticed that our query networks, i.e. the decoders, do not exploit the similarities present in demonstration trajectories. Take a sinusoidal wave of multiple cycles as an example. The CNEP model attempts to represent every single point in the trajectory individually instead of learning a single cycle and generalizing this information to the rest of the trajectory. Yet, given the frequency inside the demonstration trajectory, it should have been straightforward to discover and leverage the cyclical characteristic of the behavior. By looking at the literature, we noticed that periodic versions of MP-based modeling frameworks have been proposed as solutions to this problem. Studies like [45] proposed using nonlinear oscillators to increase the performance in modeling these behaviors. Inspired by the success of positional encoding in the recent transformer architecture in [47], we proposed the PEMP model, an MP-modeling framework, as detailed in Section 5. This model is an improved version of the CNMP model, specifically tailored with sinusoidal positional encoding for periodic behaviors. Evaluations of this model supported our hypothesis that biasing the network towards rhythmic patterns increased the modeling performance and decreased the required computational resources in periodic skills.

Considering a typical 7-DoF robotic arm or a 26-DoF robotic hand, only a subset of all DoFs exhibit true periodicity for a periodic skill. As a result, a promising research direction in the future can be integrating the linear and angular phases together to let the neural network decide which one to use for which dimension. Also, the use of PEMP as an integral part of the CNEP framework can increase the performance of the CNEP model, allowing it to use position-enhanced representations for rhythmic skills and linear representations for ordinary skills.

7. CONCLUSION

In this thesis, we propose using MP-based LfD frameworks for learning better representations about: (1) the navigation of mobile robots and (2) the manipulation of robotic arms. Learning from human demonstrators is a realistic and practical approach to advancing robotic functionality one step forward. We believe LfD is a critical paradigm aligning with our current capabilities in representing and synthesizing intelligent behavior in robots.

Section 3 presents our idea of using CNP networks to encode the social navigation behavior of humans. In Section 4, we introduce the CNEP model, which uses a MoE approach and entropy concept to learn a variety of skills simultaneously. Finally, in Section 5, we present our solution to improve the representational power of MP-based modeling frameworks in periodic skills.

Modeling the real-world data and operating in the real world brings many considerations. Nevertheless, we showed the real-world applicability of the CNEP and PEMP models. Also, we evaluated the data-driven navigation framework on realistic simulators. We believe these studies' results help advance robotic intelligence and pave the way for follow-up research with an even higher impact.

REFERENCES

1. Hirose, S., "Tenser Actuated Elastic Manipulator", *International Federation for the Promotion of Mechanism and Machine Science World Congress*, pp. 978–981, New Delhi, India, 1983.

2. Tang, W. and P. Daoutidis, "Data-driven Control: Overview and perspectives", *American Control Conference*, pp. 1048–1064, Atlanta, USA, 2022.

3. Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-Level Control Through Deep Reinforcement Learning", *Nature*, Vol. 518, No. 7540, pp. 529–533, 2015.

4. Ude, A., A. Gams, T. Asfour and J. Morimoto, "Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives", *IEEE Transactions on Robotics*, Vol. 26, No. 5, pp. 800–815, 2010.

5. Nair, A., B. McGrew, M. Andrychowicz, W. Zaremba and P. Abbeel, "Overcoming Exploration in Reinforcement Learning with Demonstrations", *International Conference on Robotics and Automation*, pp. 6292–6299, Brisbane, Australia, 2018.

6. Vygotsky, L. S., *Mind in Society: The Development of Higher Psychological Processes*, Harvard University Press, Cambridge, USA, 1978.

7. Argall, B. D., S. Chernova, M. Veloso and B. Browning, "A Survey of Robot Learning from Demonstration", *Robotics and Autonomous Systems*, Vol. 57, No. 5, pp. 469–483, 2009.

8. Hart, C. B. and S. F. Giszter, "A Neural Basis for Motor Primitives in the Spinal Cord", *Journal of Neuroscience*, Vol. 30, No. 4, pp. 1322–1336, 2010.

9. Thoroughman, K. A. and R. Shadmehr, "Learning of Action Through Adaptive Combination of Motor Primitives", *Nature*, Vol. 407, No. 6805, pp. 742–747, 2000.

10. Schaal, S., J. Peters, J. Nakanishi and A. Ijspeert, "Learning Movement Primitives", *International Symposium of Robotics Research*, pp. 561–572, Siena, Italy, 2005.

11. Ijspeert, A. J., J. Nakanishi and S. Schaal, "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots", *International Conference on Robotics and Automation*, pp. 1398–1403, Washington, USA, 2002.

12. Paraschos, A., C. Daniel, J. R. Peters and G. Neumann, "Probabilistic Movement Primitives", *International Conference on Neural Information Processing Systems*, Nevada, USA, 2013.

13. Deniša, M., A. Gams, A. Ude and T. Petrič, "Learning Compliant Movement Primitives Through Demonstration and Statistical Generalization", *Transactions on Mechatronics*, Vol. 21, No. 5, pp. 2581–2594, 2015.

14. Huang, Y., L. Rozo, J. Silvério and D. G. Caldwell, "Kernelized Movement Primitives", *The International Journal of Robotics Research*, Vol. 38, No. 7, pp. 833–852, 2019.

15. Zhou, Y., J. Gao and T. Asfour, "Learning Via-Point Movement Primitives with Inter- and Extrapolation Capabilities", *International Conference on Intelligent Robots and Systems*, pp. 4301–4308, Macau, 2019.

16. Seker, M. Y., M. Imre, J. H. Piater and E. Ugur, "Conditional Neural Movement Primitives", *Robotics: Science and Systems*, Freiburg im Breisgau, Germany, 2019.

17. Yildirim, Y. and E. Ugur, "Learning Social Navigation from Demonstrations with Conditional Neural Processes", *Interaction Studies*, Vol. 23, No. 3, pp. 427–468,

2022.

18. Yildirim, Y. and E. Ugur, "Conditional Neural Expert Processes for Learning Movement Primitives from Demonstration", *Robotics and Automation Letters*, Vol. 9, No. 12, pp. 10732–10739, 2024.

19. Sheridan, T. B., "A Review of Recent Research in Social Robotics", *Current Opinion in Psychology*, Vol. 36, pp. 7–12, 2020.

20. Fox, D., W. Burgard and S. Thrun, "The Dynamic Window Approach to Collision Avoidance", *Robotics and Automation Magazine*, Vol. 4, No. 1, pp. 23–33, 1997.

21. Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *International Conference on Robotics and Automation*, pp. 500–505, St. Louis, USA, 1985.

22. Borenstein, J., Y. Koren *et al.*, "The Vector Field Histogram: Fast Obstacle Avoidance for Mobile Robots", *Transactions on Robotics and Automation*, Vol. 7, No. 3, pp. 278–288, 1991.

23. Warren, C. W., "Global Path Planning Using Artificial Potential Fields", *International Conference on Robotics and Automation*, pp. 316–317, Scottsdale, USA, 1989.

24. Miotto, R., F. Wang, S. Wang, X. Jiang and J. T. Dudley, "Deep Learning for Healthcare: Review, Opportunities, and Challenges", *Briefings in Bioinformatics*, Vol. 19, No. 6, pp. 1236–1246, 2018.

25. Dastile, X., T. Celik and M. Potsane, "Statistical and Machine Learning Models in Credit Scoring: A Systematic Literature Survey", *Applied Soft Computing*, Vol. 91, p. 106263, 2020.

26. Dara, S., S. Dhamercherla, S. S. Jadav, C. M. Babu and M. J. Ahsan, "Machine

Learning in Drug Discovery: A Review", *Artificial Intelligence Review*, Vol. 55, No. 3, pp. 1947–1999, 2022.

27. Zhen, H., X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong and C. Gan, "3D-VLA: A 3D Vision-Language-Action Generative World Model", arXiv:2403.09631, 2024.

28. Kim, M. J., K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang and C. Finn, "OpenVLA: An Open-Source Vision-Language-Action Model", arXiv:2406.09246, 2024.

29. Pacchierotti, E., H. I. Christensen and P. Jensfelt, "Evaluation of Passing Distance for Social Robots", *International Symposium on Robot and Human Interactive Communication*, pp. 315–320, Hatfield, UK, 2006.

30. Sisbot, E. A., L. F. Marin-Urias, R. Alami and T. Simeon, "A Human-Aware Mobile Robot Motion Planner", *Transactions on Robotics*, Vol. 23, No. 5, pp. 874–883, 2007.

31. Tipaldi, G. D. and K. O. Arras, "Please Do Not Disturb! Minimum Interference Coverage for Social Robots", *International Conference on Intelligent Robots and Systems*, pp. 1968–1973, San Francisco, USA, 2011.

32. Holtz, J. and J. Biswas, "SocialGym: A Framework for Benchmarking Social Robot Navigation", *International Conference on Intelligent Robots and Systems*, pp. 11246–11252, Kyoto, Japan, 2022.

33. Zhu, K. and T. Zhang, "Deep Reinforcement Learning-Based Mobile Robot Navigation: A Review", *Tsinghua Science and Technology*, Vol. 26, No. 5, pp. 674–691, 2021.

34. Sathyamoorthy, A. J., U. Patel, M. Paul, N. K. S. Kumar, Y. Savle and

D. Manocha, “CoMet: Modeling Group Cohesion for Socially Compliant Robot Navigation in Crowded Scenes”, *Robotics and Automation Letters*, Vol. 7, No. 2, pp. 1008–1015, 2021.

35. Wang, W., R. Wang, L. Mao and B.-C. Min, “Navistar: Socially Aware Robot Navigation with Hybrid Spatio-Temporal Graph Transformer and Preference Learning”, *International Conference on Intelligent Robots and Systems*, pp. 11348–11355, Detroit, USA, 2023.

36. Girgin, T., E. Girgin, Y. Yildirim, E. Ugur and M. Haklidir, “Bidirectional Human Interactive AI Framework for Social Robot Navigation”, arXiv:2404.04069, 2024.

37. Yildirim, Y., M. Suzer and E. Ugur, “Learning Early Social Maneuvers for Enhanced Social Navigation”, arXiv:2403.15813, 2024.

38. Han, J. R., H. Thomas, J. Zhang, N. Rhinehart and T. D. Barfoot, “DR-MPC: Deep Residual Model Predictive Control for Real-world Social Navigation”, arXiv:2410.10646, 2024.

39. Canh, T. N., X. HoangVan and N. Y. Chong, “Enhancing Social Robot Navigation with Integrated Motion Prediction and Trajectory Planning in Dynamic Human Environments”, *International Conference on Control, Automation and Systems*, pp. 731–736, Jeju, Korea, 2024.

40. Narasimhan, S., A. H. Tan, D. Choi and G. Nejat, “OLiVia-Nav: An Online Lifelong Vision Language Approach for Mobile Robot Social Navigation”, arXiv:2409.13675, 2024.

41. Garnelo, M., D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende and S. A. Eslami, “Conditional Neural Processes”, *International Conference on Machine Learning*, pp. 1704–1713, Stockholm, Sweden, 2018.

42. Yildirim, Y. and E. Ugur, "Learning Social Navigation from Demonstrations with Deep Neural Networks", arXiv:2404.11246, 2024.

43. Pignat, E. and S. Calinon, "Bayesian Gaussian Mixture Model for Robotic Policy Imitation", *Robotics and Automation Letters*, Vol. 4, No. 4, pp. 4452–4458, 2019.

44. Blessing, D., O. Celik, X. Jia, M. Reuss, M. Li, R. Lioutikov and G. Neumann, "Information Maximizing Curriculum: A Curriculum-Based Approach for Learning Versatile Skills", *International Conference on Neural Information Processing Systems*, pp. 51536–51561, New Orleans, USA, 2023.

45. Ijspeert, A. J., J. Nakanishi and S. Schaal, "Learning Rhythmic Movements by Demonstration Using Nonlinear Oscillators", *International Conference on Intelligent Robots and Systems*, pp. 958–963, Lausanne, Switzerland, 2002.

46. Abu-Dakka, F. J., M. Saveriano and L. Peternel, "Periodic DMP Formulation for Quaternion Trajectories", *International Conference on Advanced Robotics*, pp. 658–663, Ljubljana, Slovenia, 2021.

47. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need", *International Conference on Neural Information Processing Systems*, pp. 6000–6010, Long Beach, USA, 2017.

48. Vougioukas, S., "Annual Review of Control, Robotics, and Autonomous Systems", *Agricultural Robotics*, Vol. 2, No. 1, pp. 365–392, 2019.

49. Zhu, F., L. Shao, J. Xie and Y. Fang, "From Handcrafted to Learned Representations for Human Action Recognition: A Survey", *Image and Vision Computing*, Vol. 55, No. 2, pp. 42–52, 2016.

50. Kruse, T., A. K. Pandey, R. Alami and A. Kirsch, "Human-Aware Robot Navigation: A Survey", *Robotics and Autonomous Systems*, Vol. 61, No. 12, pp. 1726–

1743, 2013.

51. Hall, E., *The Hidden Dimension*, Anchor Books, New York, USA, 1966.

52. Syrdal, D. S., K. L. Koay, M. L. Walters and K. Dautenhahn, "A Personalized Robot Companion-The Role of Individual Differences on Spatial Preferences in HRI Scenarios", *International Conference on Robot and Human Interactive Communication*, pp. 1143–1148, Jeju, Korea, 2007.

53. Huang, K.-C., J.-Y. Li and L.-C. Fu, "Human-Oriented Navigation for Service Providing in Home Environment", *SICE Annual Conference*, pp. 1892–1897, Taipei, Taiwan, 2010.

54. Lam, C.-P., C.-T. Chou, C.-F. Chang and L.-C. Fu, "Human-Centered Robot Navigation—Toward a Harmoniously Coexisting Multi-Human and Multi-Robot Environment", *International Conference on Intelligent Robots and Systems*, pp. 1813–1818, Taipei, Taiwan, 2010.

55. Asghari Oskoei, M., M. Walters and K. Dautenhahn, "An Autonomous Proxemic System for a Mobile Companion Robot", *International Symposium on New Frontiers in Human-Robot Interaction*, pp. 9–15, Leicester, UK, 2010.

56. Mead, R., A. Atrash and M. J. Matarić, "Proxemic Feature Recognition for Interactive Robots: Automating Metrics from the Social Sciences", *International Conference on Social Robotics*, pp. 52–61, Amsterdam, The Netherlands, 2011.

57. Helbing, D. and P. Molnár, "Social Force Model for Pedestrian Dynamics", *Physical Review E*, Vol. 51, No. 5, pp. 4282–4286, 1995.

58. Zanlungo, F., T. Ikeda and T. Kanda, "Social Force Model with Explicit Collision Prediction", *Europhysics Letters*, Vol. 93, No. 6, p. 68005, 2011.

59. Ferrer, G., A. Garrell and A. Sanfeliu, "Robot Companion: A Social-Force Based

Approach with Human-Aware Navigation in Crowded Environments", *International Conference on Intelligent Robots and Systems*, pp. 1688–1694, Tokyo, Japan, 2013.

60. Farina, F., D. Fontanelli, A. Garulli, A. Giannitrapani and D. Prattichizzo, "Walking Ahead: The Headed Social Force Model", *PLOS ONE*, Vol. 12, No. 1, pp. 1–23, 2017.

61. Kitani, K., B. Ziebart, J. Bagnell and M. Hebert, "Activity Forecasting", *Computer Vision–ECCV*, pp. 201–214, Florence, Italy, 2012.

62. Vasquez, D., B. Okal and K. O. Arras, "Inverse Reinforcement Learning Algorithms and Features for Robot Navigation in Crowds: An Experimental Comparison", *International Conference on Intelligent Robots and Systems*, pp. 1341–1346, Chicago, USA, 2014.

63. Kim, B. and J. Pineau, "Socially Adaptive Path Planning in Human Environments Using Inverse Reinforcement Learning", *International Journal of Social Robotics*, Vol. 8, No. 1, pp. 51–66, 2016.

64. Chen, Y. F., M. Everett, M. Liu and J. P. How, "Socially Aware Motion Planning with Deep Reinforcement Learning", *International Conference on Intelligent Robots and Systems*, pp. 1343–1350, Vancouver, Canada, 2017.

65. Alahi, A., K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces", *Conference on Computer Vision and Pattern Recognition*, pp. 961–971, Las Vegas, USA, 2016.

66. Gupta, A., J. Johnson, L. Fei-Fei, S. Savarese and A. Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks", *Conference on Computer Vision and Pattern Recognition*, pp. 2255–2264, Salt Lake City, USA, 2018.

67. Kothari, P., S. Kreiss and A. Alahi, "Human Trajectory Forecasting in Crowds: A Deep Learning Perspective", *Transactions on Intelligent Transportation Systems*, Vol. 23, No. 7, pp. 7386–7400, 2021.

68. Latombe, J.-C., *Robot Motion Planning*, Springer, New York, USA, 1991.

69. Mavrogiannis, C., P. Alves-Oliveira, W. Thomason and R. A. Knepper, "Social Momentum: Design and Evaluation of a Framework for Socially Competent Robot Navigation", *Transactions on Human-Robot Interaction*, Vol. 11, No. 2, pp. 1–37, 2022.

70. Yi, S., H. Li and X. Wang, "Understanding Pedestrian Behaviors from Stationary Crowd Groups", *Conference on Computer Vision and Pattern Recognition*, pp. 3488–3496, Boston, USA, 2015.

71. Stein, P., A. Spalanzani, V. Santos and C. Laugier, "Leader Following: A Study on Classification and Selection", *Robotics and Autonomous Systems*, Vol. 75, pp. 79–95, 2016.

72. Mavrogiannis, C., F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld and J. Oh, "Core Challenges of Social Robot Navigation: A Survey", *Transactions on Human-Robot Interaction*, Vol. 12, No. 3, pp. 1–39, 2023.

73. Trautman, P. and A. Krause, "Unfreezing the Robot: Navigation in Dense, Interacting Crowds", *International Conference on Intelligent Robots and Systems*, pp. 797–803, Taipei, Taiwan, 2010.

74. Vemula, A., K. Muelling and J. Oh, "Social attention: Modeling attention in human crowds", *International Conference on Robotics and Automation*, pp. 4601–4607, Brisbane, Australia, 2018.

75. Lerner, A., Y. Chrysanthou and D. Lischinski, "Crowds by Example", *Computer Graphics Forum*, Vol. 26, No. 3, pp. 655–664, 2007.

76. Pellegrini, S., A. Ess, K. Schindler and L. Van Gool, "You'll Never Walk Alone: Modeling SocialBehavior for Multi-Target Tracking", *International Conference on Computer Vision*, pp. 261–268, Kyoto, Japan, 2009.

77. Manso, L. J., P. Nuñez, L. V. Calderita, D. R. Faria and P. Bachiller, "Socnav1: A Dataset to Benchmark and Learn Social Navigation Conventions", *Data*, Vol. 5, No. 1, p. 7, 2020.

78. Yan, Z., T. Duckett and N. Bellotto, "Online Learning for Human Classification in 3D LiDAR-based Tracking", *International Conference on Intelligent Robots and Systems*, pp. 864–871, Vancouver, Canada, 2017.

79. Martin-Martin, R., M. Patel, H. Rezatofighi, A. Shenoi, J. Gwak, E. Frankel, A. Sadeghian and S. Savarese, "JRDB: A Dataset and Benchmark of Egocentric Robot Visual Perception of Humans in Built Environments", *Transactions on Pattern Analysis and Machine Intelligence*, Vol. 45, No. 6, pp. 6748–6765, 2023.

80. Karnan, H., A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. Hart, J. Biswas and P. Stone, "Socially Compliant Navigation Dataset (SCAND): A Large-Scale Dataset of Demonstrations for Social Navigation", arXiv:2203.15041, 2022.

81. Canny, J., A. Rege and J. Reif, "An Exact Algorithm for Kinodynamic Planning in the Plane", *Symposium on Computational Geometry*, pp. 271–280, Berkeley, USA, 1990.

82. Saveriano, M., F. J. Abu-Dakka, A. Kramberger and L. Peternel, "Dynamic Movement Primitives in Robotics: A Tutorial Survey", *The International Journal of Robotics Research*, Vol. 42, No. 13, pp. 1133–1184, 2023.

83. Zeestraten, M. J., I. Havoutis, J. Silvério, S. Calinon and D. G. Caldwell, "An Approach for Imitation Learning on Riemannian Manifolds", *Robotics and Automation Letters*, Vol. 2, No. 3, pp. 1240–1247, 2017.

84. Pehlivan, A. B. and E. Oztop, "Dynamic Movement Primitives for Human Movement Recognition", *Conference of the Industrial Electronics Society*, pp. 2178–2183, Yokohama, Japan, 2015.

85. Girgin, H. and E. Ugur, "Associative Skill Memory Models", *International Conference on Intelligent Robots and Systems*, pp. 6043–6048, Madrid, Spain, 2018.

86. Calinon, S., "A Tutorial on Task-Parameterized Movement Learning and Retrieval", *Intelligent Service Robotics*, Vol. 9, pp. 1–29, 2016.

87. Ugur, E. and H. Girgin, "Compliant Parametric Dynamic Movement Primitives", *Robotica*, Vol. 38, No. 3, pp. 457–474, 2020.

88. Nguyen-Tuong, D., M. Seeger and J. Peters, "Model Learning with Local Gaussian Process Regression", *Advanced Robotics*, Vol. 23, No. 15, pp. 2015–2034, 2009.

89. Deisenroth, M. P., D. Fox and C. E. Rasmussen, "Gaussian Processes for Data-Efficient Learning in Robotics and Control", *Transactions on Pattern Analysis and Machine Intelligence*, Vol. 37, No. 2, pp. 408–423, 2013.

90. Arduengo, M., A. Colomé, J. Lobo-Prat, L. Sentis and C. Torras, "Gaussian-Process-Based Robot Learning from Demonstration", *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–14, 2023.

91. Paraschos, A., C. Daniel, J. Peters and G. Neumann, "Using Probabilistic Movement Primitives in Robotics", *Autonomous Robots*, Vol. 42, No. 3, pp. 529–551, 2018.

92. Pervez, A., Y. Mao and D. Lee, "Learning Deep Movement Primitives Using Convolutional Neural Networks", *International Conference on Humanoid Robotics*, pp. 191–197, Birmingham, UK, 2017.

93. Pérez-Dattari, R. and J. Kober, "Stable Motion Primitives via Imitation and

Contrastive Learning", *Transactions on Robotics*, Vol. 39, No. 5, pp. 3909–3928, 2023.

94. Zhou, Y., J. Gao and T. Asfour, "Movement Primitive Learning and Generalization: Using Mixture Density Networks", *Robotics & Automation Magazine*, Vol. 27, No. 2, pp. 22–32, 2020.

95. Burgard, W., A. Cremers, D. Fox, D. Haehnel, G. Lakemeyer, D. Schulz, W. Steiner and S. Thrun, "Experiences with an Interactive Museum Tour-Guide Robot", *Artificial Intelligence*, Vol. 114, No. 1, pp. 3–55, 1999.

96. Thrun, S., M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy *et al.*, "Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva", *The International Journal of Robotics Research*, Vol. 19, No. 11, pp. 972–999, 2000.

97. Nourbakhsh, I., C. Kunz and T. Willeke, "The MoBot Museum Robot Installations: A Five-Year Experiment", *International Conference on Intelligent Robots and Systems*, pp. 3636–3641, Las Vegas, USA, 2003.

98. Nonaka, S., K. Inoue, T. Arai and Y. Mae, "Evaluation of Human Sense of Security for Coexisting Robots Using Virtual Reality: Evaluation of Pick and Place Motion of Humanoid Robots", *International Conference on Robotics and Automation*, pp. 2770–2775, New Orleans, USA, 2004.

99. Fong, T., I. Nourbakhsh and K. Dautenhahn, "A Survey of Socially Interactive Robots", *Robotics and Autonomous Systems*, Vol. 42, No. 3–4, pp. 143–166, 2003.

100. Kretzschmar, H., M. Spies, C. Sprunk and W. Burgard, "Socially Compliant Mobile Robot Navigation via Inverse Reinforcement Learning", *The International Journal of Robotics Research*, Vol. 35, No. 11, pp. 1289–1307, 2016.

101. Kuderer, M., H. Kretzschmar, C. Sprunk and W. Burgard, "Feature-Based Pre-

diction of Trajectories for Socially Compliant Navigation", *Robotics: Science and Systems*, pp. 193–200, Sydney, Australia, 2012.

102. Wulfmeier, M., P. Ondruska and I. Posner, "Maximum Entropy Deep Inverse Reinforcement Learning", arXiv:1507.04888, 2016.

103. Levine, S., Z. Popovic and V. Koltun, "Nonlinear Inverse Reinforcement Learning with Gaussian Processes", *International Conference on Neural Information Processing Systems*, Granada, Spain, 2011.

104. Tai, L., J. Zhang, M. Liu and W. Burgard, "Socially Compliant Navigation Through Raw Depth Inputs with Generative Adversarial Imitation Learning", *International Conference on Robotics and Automation*, pp. 1111–1117, Tokyo, Japan, 2018.

105. Che, Y., A. M. Okamura and D. Sadigh, "Efficient and Trustworthy Social Navigation via Explicit and Implicit Robot–Human Communication", *Transactions on Robotics*, Vol. 36, No. 3, pp. 692–707, 2020.

106. Koren, Y. and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation", *International Conference on Robotics and Automation*, pp. 1398–1404, Sacramento, USA, 1991.

107. Orebäck, A. and H. I. Christensen, "Evaluation of Architectures for Mobile Robotics", *Autonomous Robots*, Vol. 14, No. 1, pp. 33–49, 2003.

108. Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Networks", arXiv:1406.2661, 2014.

109. Arjovsky, M., S. Chintala and L. Bottou, "Wasserstein GAN", arXiv:1701.07875, 2017.

110. Goodfellow, I., "NIPS Tutorial: Generative Adversarial Networks", arXiv:1701.00160, 2017.

111. Burda, Y., H. Edwards, A. Storkey and O. Klimov, "Exploration by Random Network Distillation", arXiv:1810.12894, 2018.

112. Rohmer, E., S. P. N. Singh and M. Freese, "CoppeliaSim (Formerly V-REP): A Versatile and Scalable Robot Simulation Framework", *International Conference on Intelligent Robots and Systems*, pp. 1321–1326, Tokyo, Japan, 2013.

113. Nan, X. and X. Xiaowen, "Robot Experiment Simulation and Design Based on Festo Robotino", *International Conference on Communication Software and Networks*, pp. 160–162, Xi'an, China, 2011.

114. Glasius, R., A. Komoda and S. C. Gielen, "Neural Network Dynamics for Path Planning and Obstacle Avoidance", *Neural Networks*, Vol. 8, No. 1, pp. 125–133, 1995.

115. Okal, B. and K. O. Arras, "Formalizing Normative Robot Behavior", *International Conference on Social Robotics*, pp. 62–71, Kansas City, USA, 2016.

116. Kambhampati, S. and L. Davis, "Multiresolution Path Planning for Mobile Robots", *Journal on Robotics and Automation*, Vol. 2, No. 3, pp. 135–145, 1986.

117. Quinlan, S. and O. Khatib, "Elastic Bands: Connecting Path Planning and Control", *International Conference on Robotics and Automation*, pp. 802–807, Atlanta, USA, 1993.

118. Brooks, R., "A Robust Layered Control System for a Mobile Robot", *Journal on Robotics and Automation*, Vol. 2, No. 1, pp. 14–23, 1986.

119. Biswas, A., A. Wang, G. Silvera, A. Steinfeld and H. Admoni, "SocNavBench: A Grounded Simulation Testing Framework for Evaluating Social Navigation",

Transactions on Human-Robot Interaction, Vol. 11, No. 3, pp. 1–24, 2022.

120. Tsoi, N., M. Hussein, J. Espinoza, X. Ruiz and M. Vázquez, "Sean: Social Environment for Autonomous Navigation", *International Conference on Human-Agent Interaction*, pp. 281–283, Sydney, Australia, 2020.

121. Van den Berg, J., M. Lin and D. Manocha, "Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation", *International Conference on Robotics and Automation*, pp. 1928–1935, Pasadena, USA, 2008.

122. Gordon, J., W. P. Bruinsma, A. Y. K. Foong, J. Requeima, Y. Dubois and R. E. Turner, "Convolutional Conditional Neural Processes", arXiv:1910.13556, 2020.

123. Kroemer, O., S. Niekum and G. Konidaris, "A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms", *Journal of Machine Learning Research*, Vol. 22, No. 1, pp. 1395–1476, 2021.

124. Gasparetto, A. and V. Zanotto, "A New Method for Smooth Trajectory Planning of Robot Manipulators", *Mechanism and Machine Theory*, Vol. 42, No. 4, pp. 455–471, 2007.

125. Biagiotti, L. and C. Melchiorri, *Trajectory Planning for Automatic Machines and Robots*, Springer, Heidelberg, Germany, 2008.

126. Biewald, L., "Experiment Tracking with Weights and Biases", https://www.wandb.com, accessed on Jan 10, 2024.

127. University, C. M., "CMU Motion Capture Database", http://mocap.cs.cmu.edu, accessed on Jan 10, 2024.

128. Wagener, N., A. Kolobov, F. Vieira Frujeri, R. Loynd, C.-A. Cheng and M. Hausknecht, "MoCapAct: A Multi-Task Dataset for Simulated Humanoid Control", *International Conference on Neural Information Processing Systems*,

pp. 35418–35431, New Orleans, USA, 2022.

129. Cremer, S., L. Mastromoro and D. O. Popa, "On the Performance of the Baxter Research Robot", *International Symposium on Assembly and Manufacturing*, pp. 106–111, Fort Worth, USA, 2016.

130. Lee, K. M., S. Ye, Q. Xiao, Z. Wu, Z. Zaidi, D. B. DÁmbrosio, P. R. Sanketi and M. Gombolay, "Learning Diverse Robot Striking Motions with Diffusion Models and Kinematically Constrained Gradient Guidance", arXiv:2409.15528, 2024.

131. Sandler, M., A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks", *Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Los Alamitos, USA, 2018.

132. Dermy, O., M. Chaveroche, F. Colas, F. Charpillet and S. Ivaldi, "Prediction of Human Whole-Body Movements with AE-ProMPs", *International Conference on Humanoid Robots*, pp. 572–579, Beijing, China, 2018.

133. Li, G., B. He, R. Gomez and K. Nakamura, "Interactive Reinforcement Learning from Demonstration and Human Evaluative Feedback", *International Symposium on Robot and Human Interactive Communication*, pp. 1156–1162, Nanjing, China, 2018.

134. Schaal, S., *Dynamic Movement Primitives-A Framework for Motor Control in Humans and Humanoid Robotics*, chap. 23, Springer Tokyo, 2006.

135. Ude, A., B. Nemec, J. Morimoto *et al.*, "Trajectory Representation by Nonlinear Scaling of Dynamic Movement Primitives", *International Conference on Intelligent Robots and Systems*, pp. 4728–4735, Daejeon, Korea, 2016.

136. Jahn, L., F. Wörgötter and T. Kulvicius, "Comparison of Motion Encoding Frameworks on Human Manipulation Actions", *Robotics and Autonomous Systems*, Vol. 185, p. 104869, 2025.

137. Mronga, D. and F. Kirchner, "Learning Context-Adaptive Task Constraints for Robotic Manipulation", *Robotics and Autonomous Systems*, Vol. 141, p. 103779, 2021.

138. Rajeswaran, A., V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations", arXiv:1709.10087, 2017.

139. Towers, M., A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel and A. KG, "Gymnasium: A Standard Interface for Reinforcement Learning Environments", arXiv:2407.17032, 2024.

140. Pérez-Higueras, N., F. Caballero and L. Merino, "Learning Human-Aware Path Planning with Fully Convolutional Networks", *International Conference on Robotics and Automation*, pp. 5897–5902, Brisbane, Australia, 2018.

141. Shorten, C. and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning", *Journal of Big Data*, Vol. 6, No. 1, pp. 1–48, 2019.

APPENDIX A: DATASET FOR LEARNING HUMAN NAVIGATION

A.1. Real-World Data

We trained the entire model on a real-world dataset, namely SCAND [80]. This dataset contains more than 8 hours of navigation trajectories in social environments, recorded using a teleoperated robot. Along these trajectories, researchers present the raw data from several sensors, including stereo and depth cameras, lidars, GPS, and several others.

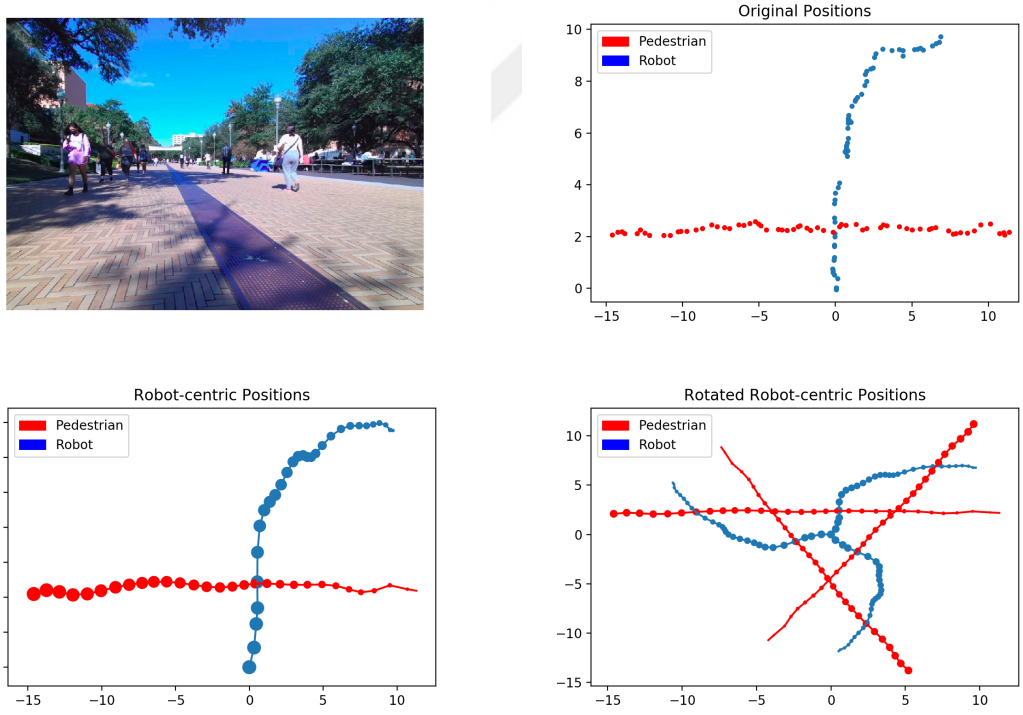


Figure A.1. Processing the data from SCAND [80]. First, raw sensor data of the selected trajectory clips with evasive maneuvers are gathered. Robot and pedestrian positions are annotated on the data and transformed into a robot-centric coordinate frame. Then, interpolation is applied to obtain entire trajectories. Finally, rotation is applied to increase the size of the dataset.

In this dataset, first, we have manually extracted the navigation data of 30 social scenarios where the robot encounters a pedestrian and avoids collision by maneuvering. In each encounter, we manually annotated the positions of the pedestrians at multiple time steps. At the same time, we obtained the robot's positions using the GPS sensor and recorded this information with corresponding time stamps. Later, we applied interpolation to position data to get the robot's and pedestrians' continuous trajectories. As in [140], where researchers use robot-centric data to learn social navigation, we processed the position data of timed trajectories to convert it to egocentric. Then, we applied rotational transformations to egocentric trajectories for data augmentation purposes. Data augmentation is the process of artificially increasing the size of the dataset. This technique has been used in many deep learning applications widely [141]. The entire data-processing procedure is depicted in Figure A.1.

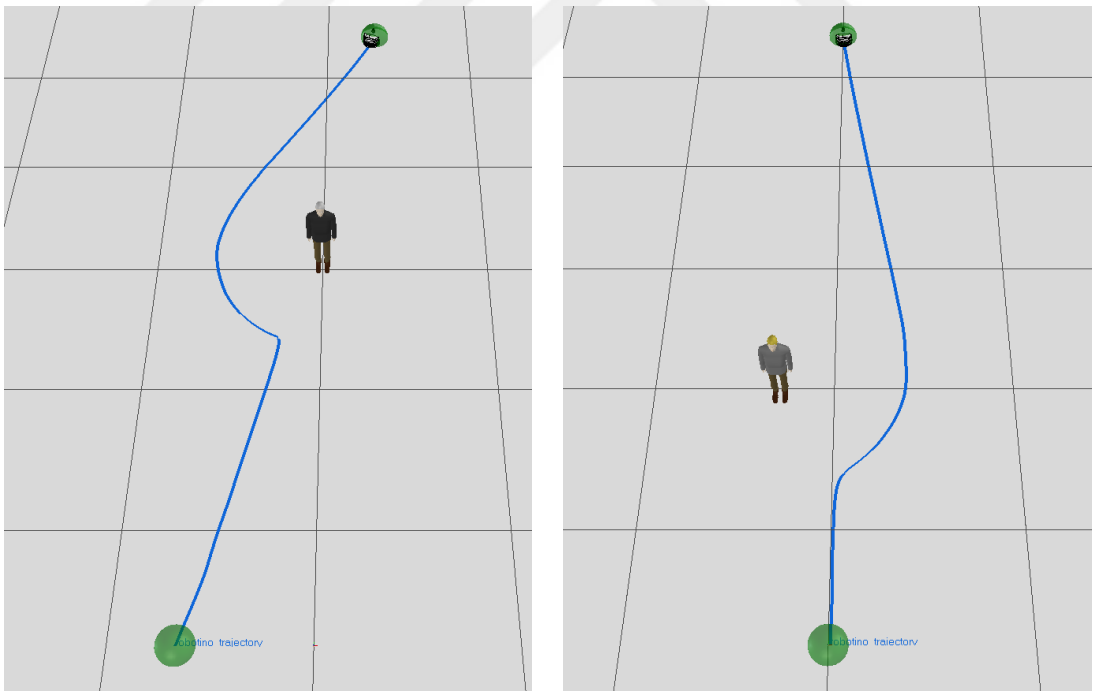


Figure A.2. Data collection on the simulation. The robot implements the Social Force Model to avoid randomly placed pedestrians. Green spheres show the navigation task's start and goal positions, and the blue line shows the motion trajectory.

A.2. Simulated Data

To gather a set of socially compliant demonstration trajectories for learning, we implemented the Social Force Model, described in [57] to control the robot. Assuming it generates socially plausible trajectories, we recorded 1500 trajectories with random start, goal, and pedestrian positions. In each trial, single and multiple stationary and dynamic pedestrians are placed randomly. Figure A.2 shows snapshots of this procedure from the simulator.

APPENDIX B: MODEL HYPER-PARAMETERS USED IN SOCIAL NAVIGATION

The implementation of the model and the training data can be found at `https://github.com/yildirimyigit/cnmp`. In addition, data-processing scripts, ROS nodes, and environments used in the simulator can be found at `https://github.com/yildirimyigit/irl_sfm`. In the following, we present the control parameters of all models used throughout the study.

Table B.1. Model parameters of SFM.

Hyperparameter	**Value**
Relaxation Time	2.3
Force Strength	6.40
Force Range	0.25
Maximum Velocity	2.5

Table B.2. Hyperparameters of the CNP of the Data-Driven Global Controller.

Hyperparameter	**Value**
Maximum Number of Observations	10
Number of Hidden Layers	3
Width of Layers of Encoder Network	256, 256, 256
Width of Layers of Query Network	256, 256, 256
Optimizer	Adam
Activation Function	ReLU
Learning Rate	1e-4

Table B.3. Hyperparameters of the CNP of the Data-Driven Local Controller.

Hyperparameter	**Value**
Maximum Number of Observations	20
Number of Hidden Layers	3
Width of Layers of Encoder Network	256, 384, 512
Width of Layers of Query Network	512, 384, 256
Optimizer	Adam
Activation Function	ReLU
Learning Rate	1e-4

Table B.4. Hyperparameters of the Feed-Forward Neural Network as the global controller.

Hyperparameter	**Value**
Number of Hidden Layers	3
Width of Hidden Layers	256, 512, 1024
Optimizer	SGD
Activation Function	ReLU
Learning Rate	1e-3

Table B.5. Hyperparameters used in the GAN of Failure Prediction Module.

Hyperparameter	**Value**
Number of Hidden Layers	2
Width of Hidden Layers	128, 128
Generator Noise	64 Dimensional, Diagonal Gaussian
Optimizer	Adam
Activation Function	ReLU
Learning Rate	1e-4

Table B.6. Hyperparameters used in the RND of Failure Prediction Module.

Hyperparameter	**Value in Target**	**Value in Predictor**
Number of Hidden Layers	3	3
Width of Hidden Layers	512,512,512	512,512,512
Optimizer	N/A	Adam
Activation Function	ELU	ELU
Learning Rate	N/A	1e-4