論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Accelerating robot learning of motor skills with knowledge transfer
著者(和文)	MAKONDONdivhuwo
Author(English)	Ndivhuwo Makondo
出典(和文)	学位:博士(学術), 学位授与機関:東京工業大学, 報告番号:甲第10900号, 授与年月日:2018年3月26日, 学位の種別:課程博士, 審査員:長谷川 修,山村 雅幸,寺野 隆雄,石井 秀明,青西 亨
Citation(English)	Degree:Doctor (Academic), Conferring organization: Tokyo Institute of Technology, Report number:甲第10900号, Conferred date:2018/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Accelerating robot learning of motor skills with knowledge transfer

by

Ndivhuwo Makondo

Submitted to the Department of Computational Intelligence and Systems Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computational Intelligence and Systems Science

at the

TOKYO INSTITUTE OF TECHNOLOGY

March 2018

Author .....
Department of Computational Intelligence and Systems Science
January 09, 2018

Certified by .....
Osamu Hasegawa
Associate Professor, Department of Computational Intelligence and Systems Science
Thesis Supervisor

Accepted by .....
Prof. Takao Terano, Prof. Masayuki Yamamura, Assoc. Prof. Hideaki Ishii, Assoc. Prof. Toru Aonishi
Thesis committee, Department of Computational Intelligence and Systems Science

Accelerating robot learning of motor skills with knowledge transfer

by

Ndivhuwo Makondo

Submitted to the Department of Computational Intelligence and Systems Science
on January 09, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computational Intelligence and Systems Science

Abstract

Machine learning approaches have recently been adopted for robot modeling and control, where robot skills are acquired and adapted from data generated by the robot while interacting with its environment through a trial-and-error process. This endows robots with the capabilities to adapt to changing and open-ended environments, by acquiring new skills and behaviors as their environments and demands change. Furthermore, as learning-based control techniques are capable of modeling complex systems, modern robots no longer have to be designed to allow modeling to be straightforward but can be designed to suit various demands and environments.

Despite the success that learning-based approaches promise us, robot learning of new control skills remains one of the main challenges, especially for manipulators and humanoids, mainly due to their large and high-dimensional state and action spaces, as a large amount of data must be collected and this requires long training times. Furthermore, their physical embodiment allows only for a limited time for collecting training data.

In this thesis, we aim to accelerate robot learning of motor skills using knowledge transfer, where we re-use data generated by other pre-existing robots to accelerate learning for a new robot. The advantage of transferring raw data is that the knowledge transfer system will not restrict the robots to the same knowledge representations and thus the same type of learning algorithms. However, knowledge transfer across robots is fraught with many challenges, specifically due to the robots having different embodiments and physical characteristics.

We propose a transfer learning model, Local Procrustes Analysis, and algorithms for training it, to enable knowledge transfer across such robots with different embodiments and physical characteristics. We demonstrate the efficacy of our proposed model in accelerating learning of manipulator kinematics and dynamics. More specifically, we accelerate learning of sensorimotor mappings – forward and inverse kinematics – and online learning of inverse dynamics for manipulator control. Moreover, we propose an approach that contributes towards robot learning from demonstrations that enables non-robotics-expert human users to transfer skills to robots, and

demonstrate that our approach can be extended to allow robots to share knowledge acquired from a human.

We validate our approach using simulated robots, ranging from simple planar manipulator robots to more complex industrial manipulators and humanoids. Our results demonstrate that not only is transfer across robots possible, but it is also beneficial to accelerating learning for new robots based on data gathered by more experienced robots.

Thesis Supervisor: Osamu Hasegawa
Title: Associate Professor, Department of Computational Intelligence and Systems Science

Acknowledgments

My PhD studies were supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) during the first three years and by Core Research for Evolutional Science and Technology (CREST) program of Japan Science and Technology Agency (JST) in the last year. I am grateful for the opportunity they allowed to me to pursue my PhD with a peace of mind, knowing my living expenses in Japan are covered.

Many thanks go to my supervisor, Associate Professor Osamu Hasegawa, who advised that I pursue my studies in the hot topic of knowledge transfer in robotics. I am grateful to all the support that he has given me during the course of my studies and stay in Japan. I would also like to acknowledge my colleague and mentor, Dr. Benjamin Rosman, who has contributed enormously to the development and completion of my thesis, and for all the fruitful discussions we had over Skype.

I would also like to thank all my lab mates at Hasegawa Laboratory, past and present, for their helpful comments and discussions during our regular seminars. Many thanks also go to the friends I made while in Japan for their moral support. I would also like to thank my family back in South Africa for their moral support and words of encouragement.

Lastly, I am grateful for the Council for Scientific and Industrial Research (CSIR), my current employer, for granting me the leave of absence to pursue my studies in Japan.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The presence of personal and service robots in real-world settings has been a long-standing vision of the artificial intelligence and robotics communities for years. This includes personal robots in homes that help with domestic tasks such as cooking, cleaning and gardening; and service robots such as those that assist doctors with surgical procedures in hospitals and those that autonomously perform duties in environments deemed unsafe for humans, such as in space.

Early approaches in pursuit of this vision were based on reasoning and human insight, where a robot engineer models a task as accurately as possible, for example, in terms of desired trajectories for a robot to follow, and uses her understanding of required forces to be applied to the robot in order to produce desired robot behaviors. Such engineered approaches in most cases are applied in pre-structured environments, for pre-defined tasks that can be fully specified mathematically and with precisely located objects. To design control policies for the robot to produce desired behaviors to complete the tasks, typically hand-crafted models based on human understanding of physics are used.

These approaches have had successes in controlled and static environments such as factories and research labs. However, they have failed to produce any desirable results in real-world environments, that are unstructured and dynamic, such as those mentioned above. In these cases, robots must be able to adapt to changing and open-ended environments, by acquiring new skills and behaviors as their environments

and demands change. To this end, data-driven approaches based on machine learning have recently been adopted. In these approaches robot skills are acquired and adapted from data generated by the robot while interacting with its environment through a trial-and-error process.

1.1 Motivation

Despite the success that learning-based approaches promise us, robot learning of new control skills remains one of the main challenges, especially for manipulators and humanoids, mainly due to their large and high-dimensional state and action spaces. Furthermore, their physical embodiment allows only for a limited time for collecting training data. For example, learning policies through reinforcement learning or developmental learning approaches generally requires an interaction of the robot with its physical environment to collect training samples over many trials.

In this thesis, we aim to accelerate robot learning with knowledge transfer, where learning of a new robot is initialized by data generated by pre-existing robots or when possible, from human demonstrations. By re-using data generated by other robots or humans as a means of knowledge transfer, the hope is that learning for the new robot will be biased towards relevant spaces such that fewer trials of interacting with the environment are needed, thus improving the learning speed.

Our motivation behind knowledge transfer is the ability of humans to retain and use knowledge acquired previously to solve new tasks faster. Humans do not only learn from their own past experiences but also from those of others by means of knowledge sharing. For example, pupils use knowledge gained from their teachers to enhance their skills in order to solve problems.

1.2 Major Contributions

Equipping robots with the capabilities to transfer knowledge that they have learned through their individual experiences is fraught with many challenges. The challenge

that we confront in this thesis is due to the diversity in modern robots, in terms of different embodiments and physical characteristics. Although this brings with it the advantage of combining the different capabilities that individual robots posses to solve complex tasks in multi-robot systems, it also makes it challenging to share *learned* knowledge across robots, because learning is often embodiment-specific.

The differences generally include kinematics properties – joint configurations, number of degrees-of-freedom (DoF), robot dimensions – and dynamic properties – mass, center of mass, inertia matrix, etc. All these result in robots having different state and action spaces, which makes transfer of low-level, embodiment-specific knowledge challenging.

This thesis makes the following contributions to the field of machine learning for robot control. We demonstrate the benefit of knowledge transfer across robots for accelerating robot learning. In particular, we demonstrate this for learning robot sensorimotor mappings, such as forward and inverse kinematics, and online learning of robot dynamics models, such as inverse dynamics. Furthermore, we also contribute an approach for humans to accelerate learning of robot motor skills, by transferring their knowledge to robots using camera systems, and demonstrate that our approach can be easily extended to transfer between robots, knowledge gained by one robot from human demonstrations.

This thesis also contributes to the field of machine learning, a model, and algorithms for training it, for knowledge transfer using non-linear manifold alignment. Although we only demonstrate the efficacy of our model in robotics applications, we believe it also has applications outside the robotics community.

Despite our contributions, challenges still remain in transferring knowledge across robots and this will be discussed in more details in Chapter 7. More specifically, when transferring kinematic data across robots, we had to rely on robot correspondences specified by a human designer. The challenge is that in some cases these correspondences may not be obvious to the human eye, especially to non-robotics-expert users, and ideally the robots should autonomously discover correspondences among themselves.

Research conducted in this thesis has led to the following papers:

- **Ndivhuwo Makondo**, Benjamin Rosman, Osamu Hasegawa, Manifold Mapping for Knowledge Transfer, International Symposium on Pedagogical Machines (workshop/not reviewed), Mar. 2015.

- **Ndivhuwo Makondo**, Benjamin Rosman, Osamu Hasegawa, Knowledge transfer for learning robot models via Local Procrustes Analysis, IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), pp. 1075-1082, Nov. 2015.

- Hiratsuka Michihisa, **Makondo Ndivhuwo**, Rosman Benjamin, Hasegawa Osamu, Trajectory Learning from Human Demonstrations Via Manifold Mapping, IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2016.

- **Ndivhuwo Makondo**, Michihisa Hiratsuka, Benjamin Rosman, and Osamu Hasegawa, A non-linear manifold alignment approach to robot learning from demonstrations, J. of Robotics and Mechatronics, Under Review.

- **Ndivhuwo Makondo**, Benjamin Rosman, Osamu Hasegawa, Accelerating model learning with inter-robot knowledge transfer, IEEE Robotics and Automation Letters, Under Review.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 provides an introduction to knowledge transfer in robot learning and also reviews related work in knowledge transfer for accelerating robot learning. We also lay out the assumptions and notation adopted throughout the thesis in Section 2.3. In Chapter 3 we present our proposed knowledge transfer model and illustrate it using a simple example in Section 3.6. The reader is recommended to go through Chapter 2 and 3 in order to

understand the rest of the document, as these two chapters contain the background knowledge.

Chapter 4, 5 and 6 can be read in any order. In Chapter 4 we present our proposed approach to learning from demonstration using camera systems. Our approach contributes towards enabling non-robotics-expert users to transfer knowledge to robots by demonstrating the tasks using camera systems such as Microsoft Kinect and motion capture suit. We demonstrate that our approach can be easily extended to allow knowledge transfer across robots for initializing a parameterized policy of robot motor skills.

In Chapter 5 we present our proposed approach to accelerating model learning with inter-robot knowledge transfer. In particular, we demonstrate how online learning of the inverse dynamics for manipulator control can benefit from knowledge transfer. Chapter 6 present our preliminary work in accelerating learning of sensorimotor mappings for developmental learning robots. In particular, we show how learning inverse kinematics for an autonomously exploring robot can benefit from knowledge transfer from an experienced robot. Lastly, we conclude the thesis and provide a discussion of future work in Chapter 7.

Chapter 2

Knowledge Transfer in Robot Learning

2.1 Introduction

Recent advances in machine learning and its application to robotics have endowed robots with the capabilities to learn and acquire knowledge from their own experiences gained by interacting with their surrounding environments. This brings with it many benefits as well as challenges. Learning from experience enables robots to continuously acquire new knowledge and adapt to changes in the robots themselves or the surrounding environment. However, in many practical cases robots learn *tabula rasa*, which results in slow learning, increasing sample complexity and sometimes making mastering difficult tasks infeasible. This is especially true for manipulators and humanoids with high-dimensional, continuous state and actions spaces.

With the proliferation of robots and the decrease in their cost, it seems reasonable to suggest that robots could benefit from communicating and effectively sharing knowledge that they have gleaned through their individual experiences. Such a multi-robot systems approach has the advantage of solving complex tasks that individual robots could otherwise not solve by themselves or would take a lot of time to solve. The communication and sharing of knowledge across robots would allow better cooperation amongst the robots and could also make it possible for robots to accelerate

each others' learning processes.

In a scenario where an old robot is to be replaced by a new model, either in an industrial setting or a domestic one (e.g. an old cleaning and cooking robot being replaced a by a newer model), rather than discarding the old robot with all the knowledge it acquired throughout its lifetime and the new robot learning from scratch, transferring some of the old robot's knowledge to the new one may help accelerate its learning process. Such bias from prior experiences is what enables humans to generalize and be able to learn to solve new problems faster.

However, such an endeavor of equipping robots with the capability to transfer knowledge to each other is fraught with challenges. One of the challenges is the lack of interoperability between software architectures used in many robotic platforms [100]. This is mainly due to compatibility issues between software frameworks, algorithms and data structures from different robot manufacturers. There have been attempts at solving this issue, such as the development of robot software formats, standards and components that if widely adopted and re-used would allow fast development of new robotic systems and communication across different robotic systems [119, 13, 14].

Examples are the Robot Operating System (ROS) [109] and the Open RObot COntrol Software (OROCOS) [15], which aim to provide abstraction of robot-specific components and provide data structures that allow distributed processes (e.g. sensor modules, actuator modules, robot systems, etc.) to communicate over a network. In other cases researchers rely on sub-optimal solutions based on developing ad-hoc custom software bridges between individual architectures [83, 123, 100].

The other, and perhaps more pressing, challenge that makes knowledge transfer across robots difficult is due to the diversity in modern robots, in terms of sensing and actuation capabilities. Although this brings with it the advantage of combining the different capabilities that individual robots posses to solve complex tasks in multi-robot systems, it also makes it challenging to share *learned* knowledge across robots, because learning is often embodiment-specific. The difference in sensors, materials and actuators used to assemble robots makes direct transfer of knowledge across robots infeasible, as this results in their corresponding machine learning models

having different feature spaces and data distributions.

Several studies have been conducted and methods proposed to deal with the problem of knowledge sharing across heterogeneous robots. One group of work seeks to share perceptual knowledge across robots with differing sensing mechanisms. Another group deals with the knowledge transfer of low-level manipulation or control skills across robots with differing embodiments and characteristics. Lastly, the third group deals with general robot knowledge sharing, combining both perceptual and manipulation skills. Generally this last group differs from the other groups in that it deals with both perceptual and manipulation skills. Furthermore, it also differs from the second group in that it deals with high-level skills and assumes the robots are capable of performing low-level skills, whereas work in the second group attempts to learn and transfer low-level skills that are often embodiment-specific.

Transferring perceptual knowledge addresses issues that are caused by robots having different sensing mechanisms, such as different camera types, characteristics and quality [67, 64, 60, 74, 65, 27], and different sensing perspectives, where cameras are placed at different spatial positions across robots [78, 74] or sharing knowledge across ground and aerial vehicles [78, 62, 66]. Approaches for enabling perceptual knowledge transfer include, among others, abstracting away raw sensory signals from different sensors into an intermediate representation [67, 62, 64, 63, 60], sharing learned classifiers [78, 66], and feature alignment for when the underlying representations differ [65]. Other methods rely on some similarity between the data from the robots and employ a *transfer risk* measure to quantify how much data from one robot to use in another robot [27].

Work that attempt to enable robots to share general knowledge deals with the knowledge of high-level tasks [147, 137, 58, 136, 138], where the knowledge is represented using a hierarchical network of skills that makes sharing of knowledge easy. The common theme in these works is representation of the knowledge in a high level of abstraction, thus abstracting away the robot-specific low-level knowledge. Key to these approaches is a knowledge representation framework and many different types of knowledge representations have been proposed, including the SysML (or

Systems Model Language) [58] or representations based on using ontologies such as KNOWROB (which uses the OWL2 ontology) [136], the IEEE standard Ontology for Robotics and Automation (ORA) [121] and many others [76, 75].

Transferring low-level manipulation/control skills on the other hand is very challenging, due to robots having different embodiments and physical characteristics, as the knowledge is mainly grounded in a robot-specific manner. The differences generally include kinematics properties – joint configurations, number of degrees-of-freedom (DoF), robot dimensions – and dynamic properties – mass, center of mass, inertia matrix, etc. All these result in robots having different state and action spaces, which makes transfer of low-level, embodiment-specific knowledge challenging. While knowledge transfer for all the different categories discussed above is important, this thesis mainly focuses on knowledge transfer for accelerating low-level control skills for new robots based on data generated by pre-existing robots. In particular, we focus on three different learning domains, learning motor skills encoded as parameterized motor primitives in joint space, online learning of inverse dynamics control and learning kinematics for autonomous robotic agents through developmental learning frameworks.

The advantage of transferring raw data is that the knowledge transfer system will not restrict the robots to the same knowledge representations and thus the same type of learning algorithms. For example, when sharing classifiers across robots as a means of knowledge transfer, both robots must use the same type of classifier [78, 66]. Moreover, transferring raw data also makes it possible to share knowledge across robots with different software architectures, as the data generated by robots is independent of the software architectures used to control the robots. In the next section we will review the literature of work in accelerating learning of motor skills with knowledge transfer.

2.2 Related Work

Knowledge transfer in machine learning has been studied under the framework of transfer learning (TL), where knowledge gained while solving prior tasks (*source tasks*) is leveraged to help improve learning a new related task (a *target task*) [101, 151]. Standard machine learning models make the strong assumption that the training and test data are drawn from the same distribution and are represented by the same feature spaces. This assumption breaks down in many practical situations, often due to the noisy and dynamic nature of the real world. Standard machine learning algorithms would typically require new training data in such real-world applications, which may be expensive to collect.

TL algorithms treat the training set, typically collected in controlled environments (e.g. collecting images of objects in a lab for learning classifiers), as the *source domain* and the test set, in the real world (e.g. using the learned classifiers in the real world), as the *target domain*, and learn to adapt models trained in the source domain to effectively apply in the target domain, or supplement target domain data with large source domain data, thus improving learning in the target domain.

A common idea in many TL methods is finding a shared (latent) feature space across domains that captures domain invariances and can be used to transfer knowledge across domains. Several TL algorithms for finding such latent spaces have been proposed, including those that find a lower-dimensional latent space using dimensionality reduction techniques, preserving statistical properties (e.g. matching underlying data distributions in the latent space) [102, 103], geometrical properties (e.g. local and global distances) [149, 150] and sometimes a combination of these properties [148].

Applications of TL can be found in computer vision problems, where TL is sometimes referred to as domain adaptation [51, 50, 44]; cross-lingual applications, where models learned in one language are transferred to another language [148, 150]; bioinformatics [149, 7]; localization [102, 103]; text classification [30, 103]; and robotics.

The majority of work applying TL to accelerate learning robot motor skills are

found in the reinforcement learning (RL) literature (see [133, 73] for surveys). In order to scale reinforcement learning to high dimensional problems with continuous state and actions spaces, such as in robot control, policy gradient methods are some of the most successful in the RL for robotics literature [106], where a parameterized policy representation is used and the policy parameters are learned by maximizing a reward function. However, as policy gradient methods are local learning methods, to learn complex movements fast the policy parameters must be properly initialized with domain knowledge [106]. Transfer learning has been employed in the RL domain to accelerate learning motor skills by initializing policy parameters for new tasks based on prior knowledge gained while solving other tasks within a single robot, the *inter-task transfer* case [134, 117, 1, 118], or for the same task for a new robot based on knowledge previously gained by other robots, the *inter-robot transfer* case [12, 72, 154].

The vast experience and task knowledge of human agents have also been leveraged as a source of knowledge to accelerate robot learning of motor skills. Transferring human knowledge onto robots has been studied in robotics under the framework of learning from demonstrations (LfD), also referred by many other terms such as imitation learning, programming by demonstration, learning by demonstrations, etc. Approaches in LfD aim to provide a user-friendly interface for non-robotics-expert users to program and transfer knowledge to robots.

LfD can be applied to accelerate learning by providing the robot with human demonstrations of the skills prior to learning in order to bias its subsequent exploration into relevant spaces such that it converges faster [106, 135]. In other cases the human agent provides guidance to the robot learner by interacting with it during its exploration phase, providing it with demonstrations of the desired skills [92]. These approaches assume the human teacher is an expert in the skills they are demonstrating to the robots. Another way to leverage human agents to accelerate robot learning, particularly for cases where expert human demonstrations are not available, is to let the human teacher provide feedback to the robot about its performance. This approach has mainly been applied to RL problems where the human feedback

is integrated into the reward functions [17, 52, 77].

In other robot learning domains such as learning internal models (e.g. kinematics and dynamics) application of transfer learning is scarce and has only recently started to surface. However, these domains face similar challenges such as long training times and the need to physically interact with the surrounding environment to collect training data. Few work has been conducted in TL for kinematics [8] and dynamics [8, 111, 110, 54] models. Our work contributes to the application of transfer learning in accelerating learning of kinematics and dynamics models.

In particular, we contribute to accelerating online learning of the inverse dynamics model for robot control and accelerating learning kinematics models of autonomous robotic agents in developmental robotics – known as learning sensorimotor mappings. Furthermore, we also contribute a data-driven transfer learning approach, based on the same ideas, to LfD, that allows an intuitive interface for humans to demonstrate tasks using camera systems to robots, without assuming knowledge of robots kinematics properties, and also show that this can also be used to transfer skills across robots. We discuss related work to each of our contributions in their relevant chapters, in Chapter 4, 5, and 6.

2.3 General Assumptions and Notation

This section serves to lay out the general assumptions made and the notation used throughout this thesis. Assumptions specific to a particular learning domain will be made explicitly in the relevant chapter. The main objective of this thesis is to investigate and design transfer learning algorithms and demonstrate the benefit of knowledge transfer in accelerating learning of robot motor skills in the learning domains introduced above.

Generally speaking, consider a standard learning agent, $\mathbf{\Omega}$, that generates a data set, $\xi = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{T}$, by interacting with its surrounding environment for a duration T in order to learn a parameterized policy $\pi_{\boldsymbol{\theta}}$. The data set ξ consists of input-output vector pairs $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$, where, for example, $\boldsymbol{x} = \boldsymbol{q} \in \Re^d$ is a joint angle vector of a

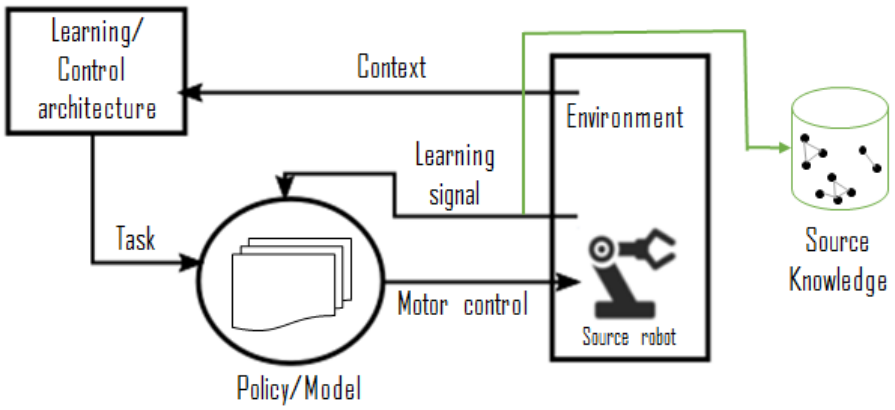


Figure 2-1: An illustration of a general learning framework. Based on an environmental context, a high-level learning architecture decides the next task that a learning agent must attempt to perform. Without prior knowledge of the task and an internal model of the agent, the agent explores its environment by executing random motor commands and observing the consequences, and use the generated data to update its policy parameters. This process repeats until the agent is competent at performing the task and has gained knowledge about the task that it will use in the future when encountering the same task.

d-DoF manipulator and $\boldsymbol{y} \in \Re^m$ is the corresponding end-effector position vector and the policy $\pi_{\boldsymbol{\theta}}$ encodes the manipulator forward kinematics function $f_{\boldsymbol{\theta}} : \boldsymbol{q} \mapsto \boldsymbol{y}$ and $\boldsymbol{\theta} \in \Theta$ are the parameters of the policy. The learner generates the data by exploring its action space and interacting with its surrounding environment, as illustrated in Fig. 2-1, and the data is used to update the policy parameters offline or online.

As a tabula rasa learning agent may learn slowly and risks damage from all the exploratory movements required to learn an optimal policy $\pi^{\star}_{\boldsymbol{\theta}}$, we aim to initialize the parameters of the target agent $\boldsymbol{\Omega}_t$ by data generated by a more experienced source agent $\boldsymbol{\Omega}_s$. The source agent can be another more experienced robot or where applicable a human teacher. We assume that the source agent is skillful in the task being learned by the target learner, i.e., the source has successfully learned the task or the human teacher is capable of generating expert demonstrations.

To this end, we want to learn the initial parameters $\boldsymbol{\theta}^{init} \approx \boldsymbol{\theta}^{\star}$ from source agent data $\xi_s = \{\boldsymbol{x}_s^{(i)}, \boldsymbol{y}_s^{(i)}\}_{i=1}^{T}$, as illustrated in Fig. 2-2. However, due to the different embodiments and characteristics between the source and target agents, we cannot use ξ_s directly to learn the initial policy $\pi_{\boldsymbol{\theta}}^{init}$ of the target agent. Thus, ξ_s must be

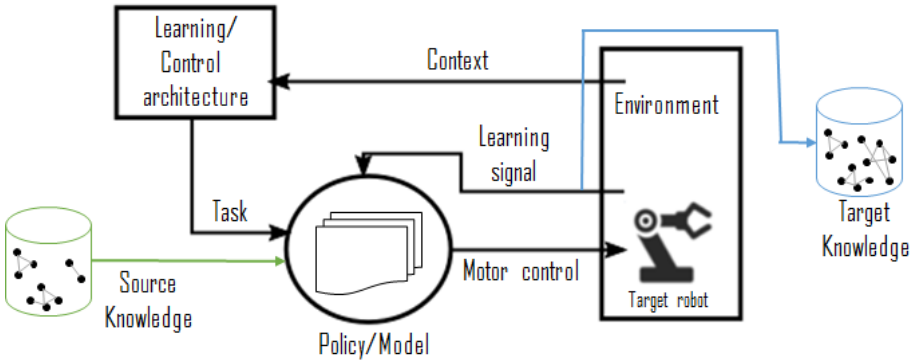


Figure 2-2: An illustration of a general transfer learning framework. Knowledge gained by a source agent when previously learning the current task is injected into the target learner policy. With the prior knowledge of the source task the agent explores its environment faster by executing motor commands that are biased toward relevant spaces.

configured such that it is useful to the target agent. For example, in our forward kinematics learning scenario, the source agent may have a different number of DoFs d_s resulting in the joint angle vector having a different dimensionality $\boldsymbol{q}_s^{(i)} \in \Re^{d_s}$ and the links of the source manipulator may have different lengths, resulting in a different distribution of its end-effector positions in the task space $\boldsymbol{y}_s^{(i)}$.

We employ an inter-robot transfer learning framework to effectively transfer the source domain knowledge to the target domain to improve learning of the target agent. We associate a source domain $\boldsymbol{\chi}_s$ with the source agent $\boldsymbol{\Omega}_s$ and a target domain $\boldsymbol{\chi}_t$ with the target agent $\boldsymbol{\Omega}_t$. We assume the domains are different but related. They are related in the sense that their data is generated by kinematic chains[1], and, for kinematics learning, they are different due to the chains having different parameters (link lengths), configurations (how the links are connected and joint offsets) and DoFs. For learning dynamics, in addition to the kinematics differences they are also different in terms of link masses, center of mass, inertia matrix, etc.

In order to effectively transfer source domain knowledge to the target domain, we learn a mapping f that maps source domain data into the target domain, i.e., $f : \boldsymbol{\chi}_s \mapsto \boldsymbol{\chi}_t$. This domain mapping f can be learned from example correspondences,

[1]A human (or robot) body can be described mathematically as a group of links (or rigid bodies) connected by joints, forming a kinematic chain.

$X_s \subset \boldsymbol{\chi}_s$ and $X_t \subset \boldsymbol{\chi}_t$, sampled from both domains; for example, by controlling the source and target agents to execute similar movements. The domain mapping f is then used to map source agent experience data ξ_s into the target domain to generate estimated target agent data $\hat{\xi}_t$, which is then used to initialize the target agent policy $\pi_{\boldsymbol{\theta}}^{init}$. As the transferred policy $\pi_{\boldsymbol{\theta}}^{init}$ may not be optimal, as the mapping is not guaranteed to be perfect, the target agent can subsequently improve this policy by trial-and-error. Fig. 2-3 illustrates our transfer learning framework.

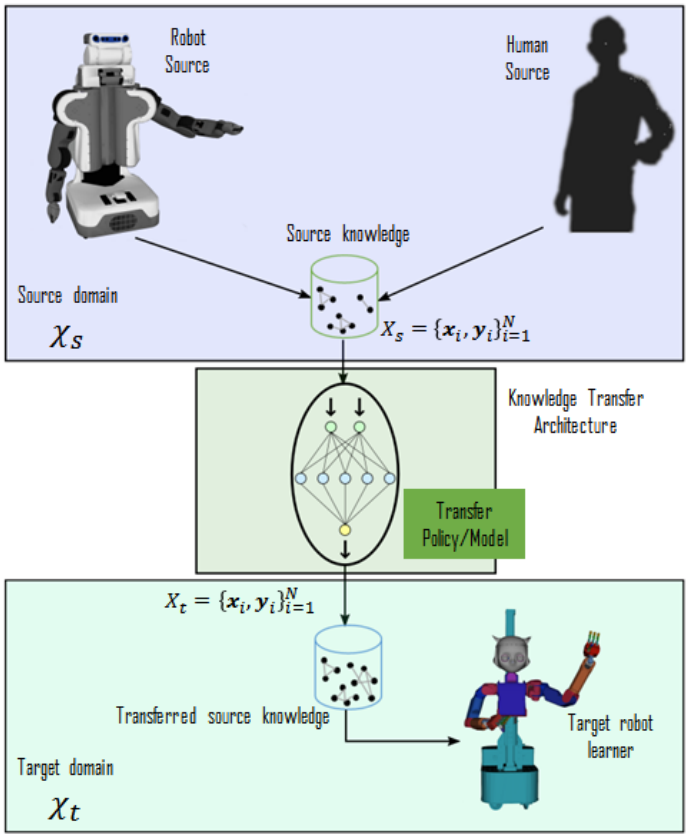


Figure 2-3: An illustration of our proposed general transfer learning framework. The framework is composed of a transfer learning model for learning the mapping f and an algorithm for generating correspondences from source and target agents.

Our transfer learning framework is composed of a transfer learning model for learning the mapping f and algorithms for generating correspondences, X_s and X_t, from the source and target agents. In Chapter 3 we discuss various models that can be employed to learn f and present our proposed model. Algorithms for generating correspondences from the agents are specific to the learning domain and thus we will

discuss these more specifically for each learning domain in Chapter 4, 5 and 6.

Chapter 3

Local Procrustes Analysis

In this chapter we present our proposed transfer learning model that is based on manifold alignment, Local Procrustes Analysis (LPA). We first provide a brief introduction to knowledge transfer with manifold alignment techniques in Section 3.1, then present the Procrustes Analysis (PA) technique, a linear manifold alignment technique upon which LPA is based, in Section 3.2. We provide an overview of LPA in Section 3.3 and sections 3.4 - 3.5 discuss various components of LPA. Lastly, we will provide a simple illustrative example showcasing the performance of LPA compared to the linear Procrustes Analysis.

3.1 Introduction

Consider two different but related domains: the source domain $\boldsymbol{\chi}_s \subset \Re^{d_s+m_s}$ and the target domain $\boldsymbol{\chi}_t \subset \Re^{d_t+m_t}$, associated with source agent $\boldsymbol{\Omega}_s$ and target agent $\boldsymbol{\Omega}_t$ respectively, as previously defined in Section 2.3. $d_j + m_j$ is the dimensionality of domain j with d_j being the dimensionality of its input space and m_j the dimensionality of its output space. In general, the dimensionality of the domains are not the same, due to the agents having different embodiments, i.e., $d_s \neq d_t$ and $m_s \neq m_t$. The objective is to find a mapping, $f : \boldsymbol{\chi}_s \mapsto \boldsymbol{\chi}_t$, that maps data points from the source domain to the target domain, through which knowledge can be shared across the domains.

Manifold alignment techniques have been shown to be very useful in this kind of problem because they allow for knowledge transfer between two seemingly disparate data sets, by aligning their underlying manifolds given a set of corresponding samples $X_s = \{\boldsymbol{x}_s^{(i)}, \boldsymbol{y}_s^{(i)}\}_{i=1}^N$ and $X_t = \{\boldsymbol{x}_t^{(i)}, \boldsymbol{y}_t^{(i)}\}_{i=1}^N$ [148, 149], where $X_s^{(i)} \in \boldsymbol{\chi}_s$ and $X_t^{(i)} \in \boldsymbol{\chi}_t$ are in correspondence. Applications of manifold alignment include automatic machine translation [37], cross-lingual information retrieval [148, 149], transfer learning for Markov Decision Processes [148, 2] and robot model learning [8], object pose alignment [153, 29] and bioinformatics [148, 149, 29]. Learning the transfer model f with manifold alignment can be accomplished using two general methods: two-step alignment methods and one-step alignment methods.

In the first step of a two-step alignment method, latent representations of the source and target data are found independently in a lower dimensional space using dimensionality reduction. In the second step, a transformation between the two is computed by aligning them in the latent space. A one-step alignment method on the other hand combines the two steps into one single step, where the data sets are projected into a shared latent space. The output of this process is two mappings that map the data sets into the shared latent space. For knowledge transfer to be possible in both methods the mappings between the original space and the latent space must be bijective, as the inverses are needed to map back to the the original spaces.

Examples of the two-step approach in robotics include combining Principal Component Analysis (PCA) and Procrustes Analysis (PA) for transfer learning of forward kinematics and inverse dynamics [8]. Examples of the one-step approach include Unsupervised Manifold Alignment (UMA) [149, 2] and shared Autoencoders [53] for transfer learning in the reinforcement learning domain, and shared Gaussian Process Latent Variable Models (shared GPLVMs) [41] for transferring kinematic data for humanoid bi-manual tasks [35]. In one-step alignment approaches, typically the dimensionality reduction is performed in a supervised manner, i.e., considering the purpose of alignment, by imposing constraints on the cost function, and linear [149, 2] and non-linear [41, 35, 53] dimensionality reduction techniques are employed.

One-step alignment approaches on the other hand perform dimensionality reduc-

tion in an unsupervised manner, and as a result the domains are not guaranteed to be the same in the latent space and an additional transformation step is required to align the domains. Although this approach does not guarantee optimal alignment of the domains in the latent space, it allows the user the choice of dimensionality reduction techniques depending on the problem at hand. For example, in some cases it may be obvious to the user how the domains are related, and thus the user can manually match the dimensionality of the domains, e.g., by removing some dimensions. Furthermore, the simple combination of PCA and Procrustes Analysis is data efficient and has been shown to accelerate learning from very few training data [8] and we also demonstrate this data efficiency in Chapter 5. However, sometimes the linear mapping of Procrustes Analysis is too restrictive in some problems. Thus, we propose Local Procrustes Analysis as an extension to Procrustes Analysis to relax the linearity assumption and handle non-linear mappings.

3.2 Procrustes Analysis

In this section we present the Procrustes Analysis technique for manifold alignment before presenting our extension in the following sections. The goal of Procrustes Analysis is to find an optimal alignment from some source data set $Z_s \subset \Re^d$ to some target data set $Z_t \subset \Re^d$, where both data sets are assumed to have the same dimensionality d, given that $Z_s^{(i)}$ is in correspondence with $Z_t^{(i)}$. Through this linear transformation, novel points in the source domain can be mapped onto the target domain. The optimal alignment is achieved by removing the translational, rotational and scaling components from one data set such that the two data sets are optimally aligned [148].

The first step in applying PA is to preprocess the data by subtracting the mean and whitening it, thus obtaining standardized matrices M_s and M_t, as follows,

$$\boldsymbol{s} = B_s(\boldsymbol{z}_s - \boldsymbol{\omega}_s), \tag{3.1}$$

$$\boldsymbol{t} = B_t(\boldsymbol{z}_t - \boldsymbol{\omega}_t), \tag{3.2}$$

where $\boldsymbol{s} \in M_s$ and $\boldsymbol{t} \in M_t$. The values $\boldsymbol{\omega}_s = \mathbb{E}\{Z_s\}$ and $\boldsymbol{\omega}_t = \mathbb{E}\{Z_t\}$ are the means of the data, where $\mathbb{E}\{\cdot\}$ denotes the expectation operator. Matrices B_s and B_t can be obtained using the Singular Value Decomposition (SVD) of the covariance matrices of Z_s and Z_t respectively, and are such that the data M_s and M_t are whitened.

The manifold alignment function is modeled as a linear mapping $f_d : M_s \mapsto M_t$, with

$$f_d(\boldsymbol{s}) = A\boldsymbol{s} \tag{3.3}$$

where $A^{d \times d}$ is a transformation matrix. To find the optimal transformation A, Wang and Mahadevan [148] minimized $\|M_t - f_d(M_s)\|_F$, where $\|\cdot\|_F$ is the Frobenius norm. Bócsi et al. [8] minimized the expected loss of the transformation instead, as described below.

We find the parameters A from (3.3) such that the expectation of the error of the transformation $L(A)$ is minimized, i.e.,

$$A = \arg\min_r L(A) \tag{3.4}$$

with

$$\begin{aligned} L(A) &= \mathbb{E}\{(\boldsymbol{t} - A\boldsymbol{s})^T(\boldsymbol{t} - A\boldsymbol{s})\} \\ &= tr(\boldsymbol{\Sigma}_{tt} - 2A^T\boldsymbol{\Sigma}_{ts} + A^T\boldsymbol{\Sigma}_{ss}A) \end{aligned} \tag{3.5}$$

where $\boldsymbol{\Sigma}_{ss}$, $\boldsymbol{\Sigma}_{tt}$ and $\boldsymbol{\Sigma}_{ts}$ are covariance matrices and L is a loss function. The minimization can be performed by setting the derivative of $L(A)$ to zero. After differentiation,

$$0 = -2\boldsymbol{\Sigma}_{ts} + 2A^T\boldsymbol{\Sigma}_{ss}$$

$$A = \boldsymbol{\Sigma}_{ss}^{-1}\boldsymbol{\Sigma}_{ts}. \tag{3.6}$$

where $\boldsymbol{\Sigma}_{ss}$ is the covariance matrix of the source matrix M_s and $\boldsymbol{\Sigma}_{ts}$ is the covariance between the source and target matrices M_s and M_t.

To find a latent space of dimensionality d and match the dimensions of the data sets, $Z_s \subset \Re^{d_s}$ and $Z_t \subset \Re^{d_t}$, for the general case where $d_s \neq d_t$, and obtain $M_s \subset \Re^d$ and $M_t \subset \Re^d$, any dimensionality reduction technique can be employed depending on

the problem, provided the inverse of the resulting mapping is well defined. Combining Procrustes Analysis with PCA is straightforward, and is achieved by separately projecting Z_s and Z_t onto their d principal components using only the first d columns of B_s and B_t respectively in 3.1 and 3.2. The first d columns are chosen such that the variances in Z_s and Z_t are maximized. The only open parameter is the dimensionality of the latent space d. Algorithm 1 summarizes the learning procedure of Procrustes Analysis with PCA.

Algorithm 1 Procrustes Analysis: Learning

1: IN: Training sets Z_s, Z_t and latent space dimension d
2: Compute M_s and M_t (using PCA, Eq. 3.1 and 3.2)
3: Compute alignment matrix: $A = \mathbf{\Sigma}_{ss}^{-1}\mathbf{\Sigma}_{ts}$
4: OUT: Parameters $\Phi = \{A, B_s, B_t, \boldsymbol{\omega}_s, \boldsymbol{\omega}_t\}$

A new point $\boldsymbol{s}_\star = B_s(\boldsymbol{z}_s^\star - \boldsymbol{\omega}_s)$ in the source manifold can then be mapped to the target manifold using $\hat{\boldsymbol{z}}_t^\star = {B_t}^{-1} A \boldsymbol{s}_\star + \boldsymbol{\omega}_t$, where $\hat{\boldsymbol{z}}_t^\star$ is the transferred point. This is summarized in Algorithm 2.

Algorithm 2 Procrustes Analysis: Transfer

1: IN: Parameters $\Phi = \{A, B_s, B_t, \boldsymbol{\omega}_s, \boldsymbol{\omega}_t\}$
2: IN: Source domain novel point $\boldsymbol{z}_s^\star$
3: Compute standardized novel point: $\boldsymbol{s}_\star = B_s(\boldsymbol{z}_s^\star - \boldsymbol{\omega}_s)$
4: Compute estimated target point: $\hat{\boldsymbol{z}}_t^\star = {B_t}^{-1} A \boldsymbol{s}_\star + \boldsymbol{\omega}_t$
5: OUT: $\hat{\boldsymbol{z}}_t^\star$

3.3 Overview of Local Procrustes Analysis

This section provides the overview of Local Procrustes Analysis. LPA extends the Procrustes Analysis method to handle non-linear mappings, by approximating a global non-linear manifold alignment with locally linear functions. This idea of approximating a global non-linear function with locally linear models has also been applied in locally weighted learning for control [4, 140] and learning non-linear image manifolds [145].

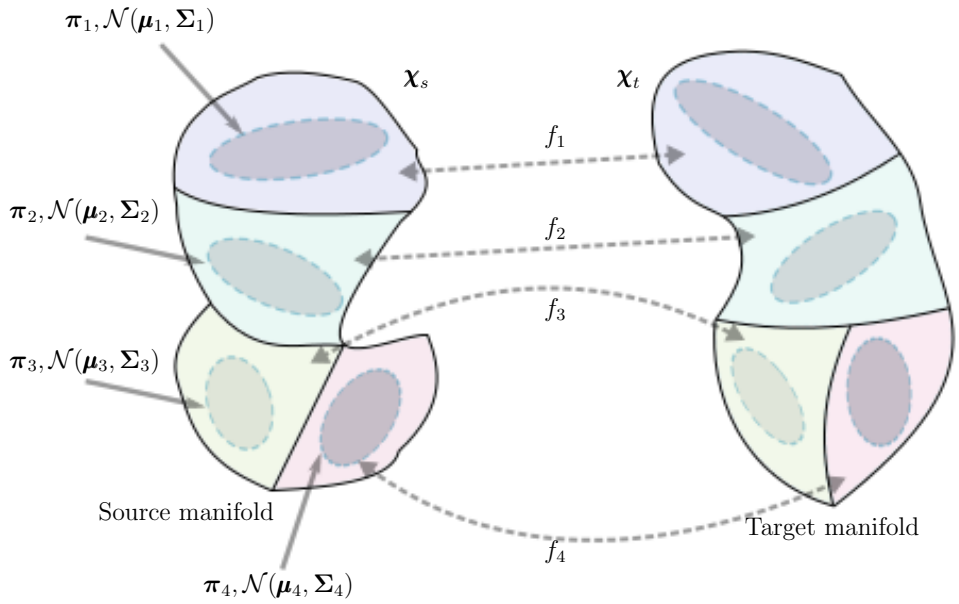


Figure 3-1: Illustration of LPA with 4 clusters. The gray shaded ellipses represent GMM components and the shaded areas on the manifolds represent areas for which the GMM components are responsible. f_1 to f_4 are linear mappings learned from the data in their corresponding clusters.

Consider again that we are provided with some source data set $X_s \subset \boldsymbol{\chi}_s$, target data set $X_t \subset \boldsymbol{\chi}_t$ and $X_s^{(i)}$ is in correspondence with $X_t^{(i)}$. The objective of LPA is to learn the domain mapping f as a non-linear function. LPA assumes that the domains are locally continuous and smooth, and that the mapping can be computed locally using linear models on the corresponding instances. To achieve this, LPA first clusters the two data sets into K local clusters. Then a linear mapping for each cluster is computed using the Procrustes Analysis algorithm. This is illustrated in Fig. 3-1 for four clusters obtained using a Gaussian Mixture Model (GMM). A new data point $\boldsymbol{z}_s^{(i)}$ from the source domain can then be mapped to the target domain by a weighted sum of the linear mappings, to obtain the estimated target point $\hat{\boldsymbol{z}}_t^{(i)}$, as follows,

$$\hat{\boldsymbol{z}}_t^{(i)} = \frac{\sum_{k=1}^{K} w_k(\boldsymbol{z}_s^{(i)}) f_k(\boldsymbol{z}_s^{(i)})}{\sum_{j=1}^{K} w_j(\boldsymbol{z}_s^{(i)})}, \tag{3.7}$$

where $w_k(\boldsymbol{z})$ is a weight that determines the influence linear mapping f_k has on data point $\boldsymbol{z}$.

We describe the formulation of LPA, its training procedure for obtaining local

clusters and its transfer algorithm in Section 3.4. Then we present one approach for initializing the training of LPA in Section 3.5.

3.4 Clustering and Mapping

This section describes how the two data sets can be clustered such that the weighted sum of the linear mappings, learned on the resulting clusters, yields a good non-linear mapping from the source domain to the target domain. The aim is to represent the two data sets X_s and X_t by a mixture of K regions, where corresponding points in the data sets map to the same region, as illustrated in Fig. 3-1. One way to obtain clusters is to use the standard unsupervised learning framework, where any standard clustering technique can be used. Clustering could be performed in one of the domains and then cluster assignments are transferred to the other domain using the correspondence information, i.e. $X_s^{(i)}$ and $X_t^{(i)}$ will share the same cluster assignment across the domains.

We choose to employ Gaussian Mixture Modeling (GMM), where the Gaussian mixtures correspond to the local regions, trained using the Expectation–Maximization (EM) algorithm, because it allows interpolating the output of local mappings using component responsibilities as weights. A GMM is represented by three parameters: the mixing coefficients π_k, the mean vectors $\boldsymbol{\mu}_k$ and the covariance matrices $\boldsymbol{\Sigma}_k$. The total probability density over a vector $\boldsymbol{x}$ is then defined as a superposition of K Gaussian densities of the form

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{3.8}$$

and the components' responsibilities are defined as

$$\gamma_k = \frac{\pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{3.9}$$

where $\mathcal{N}$ is a multivariate normal distribution. γ_k can be viewed as the responsibility

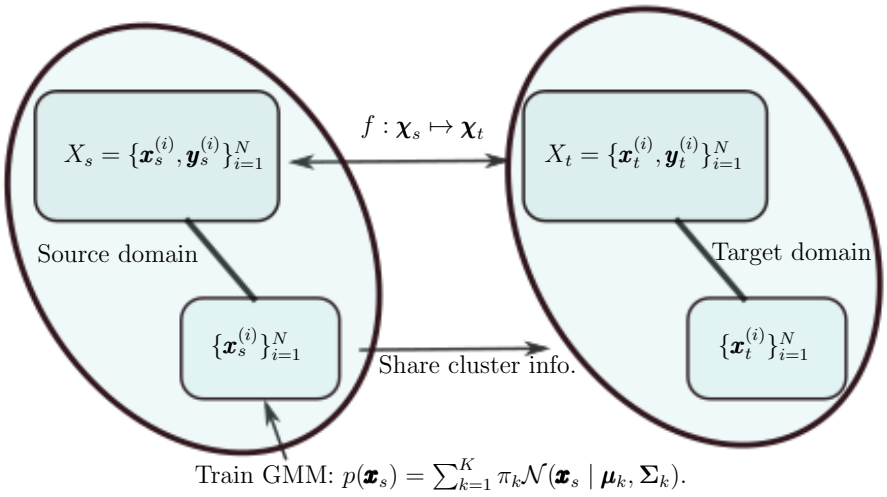


Figure 3-2: Illustration of clustering in the input space. A GMM model is trained in the input space of the source domain, and the clusters found are transferred to the input space of the target domain, using the correspondence information. The non-linear mapping f is learned in the combined spaces.

that component k takes for explaining the point $\boldsymbol{x}$.

We train a GMM on only one of the domains (the source domain throughout the thesis) and the data is clustered by assigning points to components with the highest responsibilities. This is indicated in Fig. 3-1 by having GMM parameters only in the source domain. This clustering information, together with the information about correspondences, is then used to fit another GMM to the target domain, i.e., points in the target domain that correspond to points in the same cluster in the source domain share the same cluster assignments.

Furthermore, since our domain data consists of inputs $\boldsymbol{x}_s^{(i)}$ and outputs $\boldsymbol{y}_s^{(i)}$ for learning regression models, we expect correlations between the input and output spaces. For example, in our forward kinematics learning scenario, the joint space and the end-effector space are correlated through the kinematics of the body (i.e. moving arm joints affects the end-effector movement). So in order to efficiently obtain clusters, the GMMs are trained in the input spaces of the data sets (see Fig. 3-2), or the user can specify a subspace with the most clustering information.

To learn the GMM parameters $\Pi = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ from the data X_s using the EM algorithm, we must determine the desired number of GMM components K and initial-

ize the parameters Π^{init}. We do not put any restrictions on the choice of determining the initial parameters. We, however, propose our own supervised EM initialization scheme that considers the purpose of aligning the two data sets, X_s and X_t, when clustering. This initialization scheme is described in the next section.

Given the parameters of the GMM learned in the source domain, mapping a source data point to the target domain using Eq. 3.7 becomes

$$\hat{\boldsymbol{z}}_t = \sum_{k=1}^{K} \gamma_k f_k(\boldsymbol{z}_s), \tag{3.10}$$

where we use the GMM components' responsibilities γ_k as weights, computed in the input space using Eq. 3.9 as

$$\gamma_k = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_s \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_s \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{3.11}$$

We can now use the cluster assignments obtained by assigning points in X_s and X_t to GMM components with the highest responsibility γ_k, and the Procrustes Analysis to compute f_k locally in each cluster. Algorithm 3 summarizes the steps for training LPA.

Algorithm 3 Local Procrustes Analysis: Learning

1: IN: Training sets X_s, X_t and latent space dimensionality d
2: IN: Parameters c_{min} and N_{min} {see Alg. 5 in Section 3.5}
3: (Π^{init}, K) $\Leftarrow$ initEM(X_s, X_t, c_{min}, N_{min}, d) {see Alg. 5 in Section 3.5}
4: $\Pi \Leftarrow$ fitGMM($\{\boldsymbol{x}_s^{(i)}\}_{i=1}^{N}$, Π^{init}, K)
5: **for** each cluster $k \in [1, K]$ **do**
6: Compute $\{A^{[k]}, B_s^{[k]}, B_t^{[k]}, \boldsymbol{\omega}_s^{[k]}, \boldsymbol{\omega}_t^{[k]}\}$ {see Alg. 1}
7: **end for**
8: $\Phi = \{A^{[k]}, B_s^{[k]}, B_t^{[k]}, \boldsymbol{\omega}_s^{[k]}, \boldsymbol{\omega}_t^{[k]}\}_{k=1}^{K}$
9: OUT: {Π, Φ}

First, the dimensionality of the latent space must be specified. However, if training sets X_s and X_t have the same dimensionality the parameter d can be omitted. Line 2 and 3 in Alg. 3 will be explained in Section 3.5 but they serve to initialize the EM algorithm, for fitting the GMM in Line 4. Note that as previously discussed, the

GMM is fitted in the input space $\{\boldsymbol{x}_s^{(i)}\}_{i=1}^N$. In Line 5-7 we loop through the clusters and compute linear mappings for each cluster using Alg. 1, where all the linear mappings are learned in the same latent space with dimensionality d. The output of the algorithm is the GMM parameters $\Pi = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ and the parameters of all the K linear mappings $\Phi = \{A^{[k]}, B_s^{[k]}, B_t^{[k]}, \boldsymbol{\omega}_s^{[k]}, \boldsymbol{\omega}_t^{[k]}\}_{k=1}^K$.

Algorithm 4 Local Procrustes Analysis: Transfer

1: IN: LPA parameters $\{\Pi, \Phi\}$
2: IN: Novel source point $\boldsymbol{z}_s^\star$
3: Compute each component's responsibility γ_k in input space $\boldsymbol{x}_s^\star$ {see Eq. 3.11}
4: Compute estimated target point $\hat{\boldsymbol{z}}_t^\star$ using Eq. 3.12
5: OUT: $\hat{\boldsymbol{z}}_t^\star$

Algorithm 4 summarizes the steps for knowledge transfer with LPA. For a novel data point $\boldsymbol{z}_s^\star = \{\boldsymbol{x}_s^\star, \boldsymbol{y}_s^\star\}$, $\boldsymbol{z}_s^\star \in \chi_s$ in the source domain to be transferred to the target domain, we compute each Gaussian component's weights using Eq. 3.11. Then, we map the query point $\boldsymbol{z}_s^\star$ as follows

$$\hat{\boldsymbol{z}}_t^\star = \sum_{k=1}^{K} \gamma_k ({B_t^{[k]}}^{-1} A^{[k]} \boldsymbol{z}_s^\star + \boldsymbol{\omega}_t^{[k]}). \tag{3.12}$$

3.5 Initializing the E-M Algorithm for LPA

In this section we describe our proposed EM initialization scheme for Local Procrustes Analysis. Our aim is to partition the two data sets X_s and X_t into K clusters such that a weighted sum of the linear mappings learned in each cluster approximate a good global non-linear mapping. One way to measure how good a candidate non-linear mapping is, is to compare $\hat{\boldsymbol{z}}_t^{(i)}$ in Eq. 3.7 to its corresponding ground-truth $\boldsymbol{z}_t^{(i)}$. For N training points we have the squared error cost function

$$J = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\boldsymbol{z}}_t^{(i)} - \boldsymbol{z}_t^{(i)}\|^2. \tag{3.13}$$

Substituting $\hat{\boldsymbol{z}}_t^{(i)}$ with $\boldsymbol{z}_s$ in Eq. 3.13 results in

$$J = \frac{1}{N}\sum_{i=1}^{N} \| \frac{\sum_{k=1}^{K} w_k(\boldsymbol{z}_s^{(i)}) f_k(\boldsymbol{z}_s^{(i)})}{\sum_{j=1}^{K} w_j(\boldsymbol{z}_s^{(i)})} - \boldsymbol{z}_t^{(i)} \|^2. \tag{3.14}$$

Optimizing J directly in Eq. 3.14 is computationally intractable for large K. Therefore, we must approximate it. We employ a decisive hierarchical clustering scheme, that starts with one cluster and recursively splits it until some criteria met. We introduce two thresholds as our criteria, N_{min} and c_{min}. N_{min} is the minimum number of points allowed in each cluster and c_{min} is the minimum error of each linear mapping in the clusters. We introduce the cluster mapping error c_k as follows

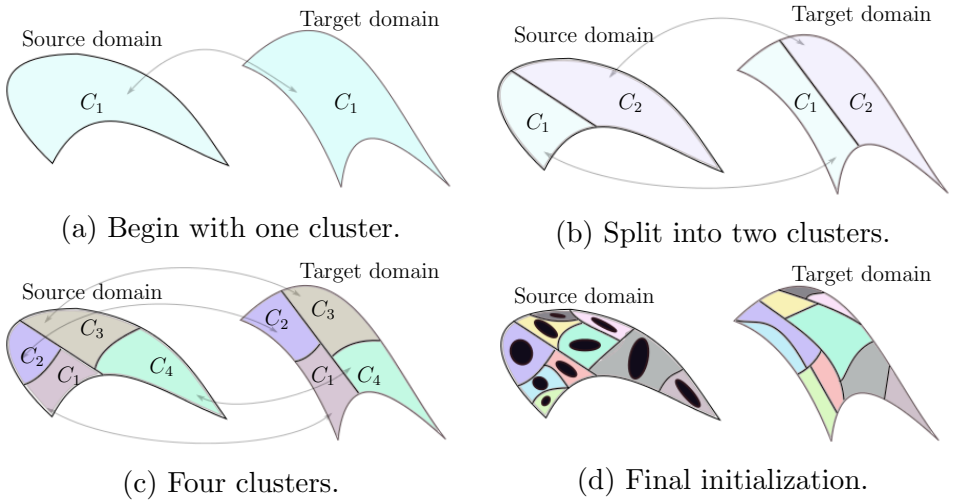
$$c_k = \frac{1}{N_k}\sqrt{\sum_{i=1}^{N_k} \| f_k(\boldsymbol{z}_s^{(i)[k]}) - \boldsymbol{z}_t^{(i)[k]} \|^2}, \tag{3.15}$$

where N_k is the number of points in cluster C_k and $\boldsymbol{z}_s^{(i)[k]}$ and $\boldsymbol{z}_t^{(i)[k]}$ are source and target points in correspondence in cluster C_k.

Algorithm 5 initEM

```
IN: Training sets X_s, X_t
IN: Parameters c_min, N_min and latent space dimensionality d
Set cluster assignment vector h to ones and K = 1
while not terminated do
    for each cluster k ∈ [1, K] do
        Learn f_k using Alg. 1
        Compute cluster error c_k (see Eq. 3.15)
        if c_k > c_min then
            Split cluster C_k into two (C_k,1 and C_k,2)
            if N_C_k,1 ≥ N_min and N_C_k,2 ≥ N_min then
                Update assignment vector h accordingly
            end if
        end if
    end for
    Update number of components K
end while
Initialize GMM components' parameters {π_k, μ_k, Σ_k}
OUT: K, Π^init = {π_k, μ_k, Σ_k}
```

This EM initialization procedure is illustrated in Fig. 3-3 and summarized in Alg.



(a) Begin with one cluster.

(b) Split into two clusters.

(c) Four clusters.

(d) Final initialization.

Figure 3-3: Illustration of the EM initialization procedure. In the final stage a GMM is learned in the source domain and the dark ellipses in 3-3d represent its Gaussian components.

5. In Line 3 the entire data is treated as one cluster, as illustrated in Fig. 3-3a. In Lines 4-15 the clusters are recursively split (see Fig. 3-3b and 3-3c) until their mapping errors c_k reach the threshold c_{min} (Line 8) and each cluster created has enough points (Line 10). We use the K-means algorithm to split clusters into two. If either of the resulting two clusters have less than N_min points, the cluster split is reversed. Lastly, in Line 17 a GMM is initialized using the final cluster assignments obtained, as illustrated in Fig. 3-3d.

The threshold c_{min} reflects the desired mapping error that the user is willing to tolerate. If set to a very small value (e.g., 10^{-3}) every cluster created will be split until each c_k reaches that threshold while maintaining $N_k \geq N_{min}$. The threshold N_{min} must have the lower bound $N_{min} \geq d$, in order to compute covariance matrices $\mathbf{\Sigma}_{ss}$ and $\mathbf{\Sigma}_{ts}$ for computing A in Eq. 3.6. For small values of c_{min}, N_{min} controls the amount of clusters created and if set to a small value it may result in overfitting, especially for small N. Alternatively, one may choose to instead specify the desired maximum number of clusters K_{max} and Alg. 5 will split the clusters until either the threshold c_{min} is reached or K_{max} clusters have been created.

Since c_{min} has no effect if set very small, we can keep it at a small value and control the number of clusters using $N_{min} \geq d$. Thus the only open parameters that

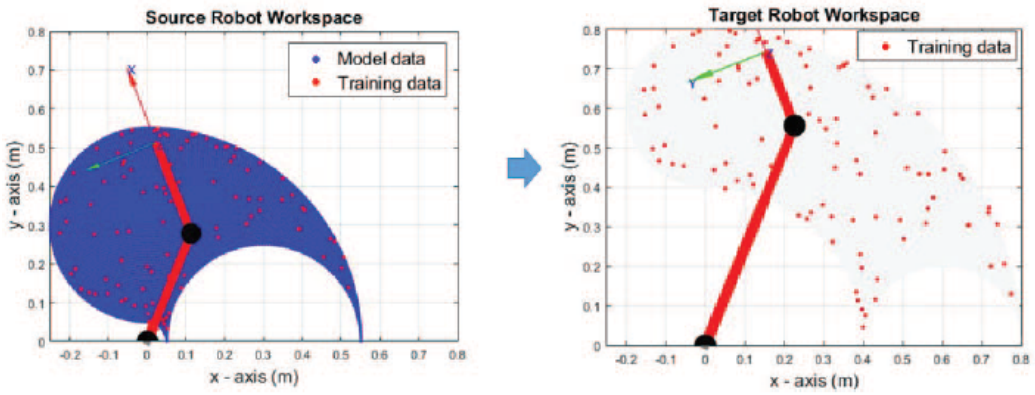


Figure 3-4: An illustrative example of knowledge transfer between robots. The source robot has collected sufficient data (blue) to learn its models, whereas the target robot has only sparse data (red). The mapping function is learned using the corresponding training data (red) from both robots. The light gray background in the target space is the workspace of the target robot to be inferred. The dark cylinders are the joints – with the base at (0,0) – connecting the two red links.

must tuned in LPA are N_{min} and the latent space dimensionality d. The output of this procedure is the number K of GMM components and the parameters $\Pi^{init} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$. In practice, Alg. 5 can be run multiple times and parameters Π^{init} with the highest log-likelihood estimate are used to initialize the EM algorithm, which is run until convergence. After the EM algorithm has converged, clusters are created locally by assigning points to components with the highest responsibilities.

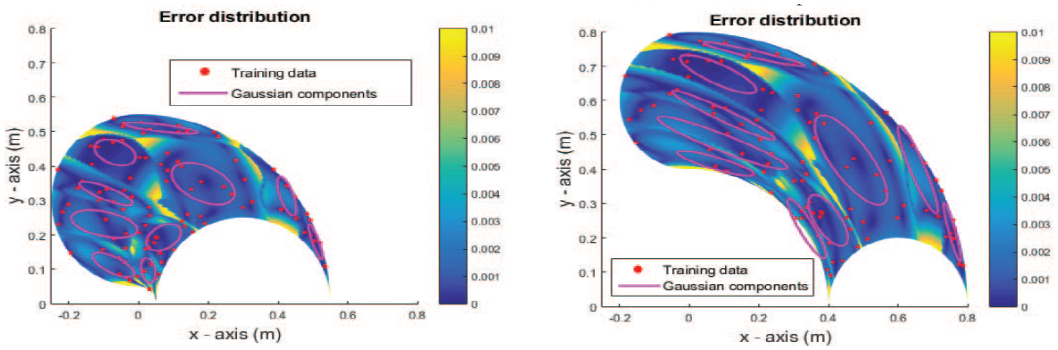
3.6 Illustrative Example

In this illustrative example we illustrate knowledge transfer on 2-DoF robots. We compare Local Procrustes Analysis against Procrustes Analysis. Note that the visualization is done in a 2D end-effector workspace and the actual datasets are 4 dimensional, i.e., $d_s = d_t = 2$ and $m_s = m_t = 2$. We compute a mapping between two planar robots, from a small set of corresponding points, for speeding up learning of forward kinematics on the target robot. Both robots have two links and two DoFs as shown in Fig. 3-4. The parameters of the two robots are shown in Table 3.1.

We performed two experiments: in experiment 1, the robots have link lengths as shown in Table 3.1; and in experiment 2, we varied the link lengths of the target robot

Parameter	Source	Target
`Link 1`	0.3	0.6
`Link 2`	0.25	0.2
`Motor 1`	$[0, \pi/2]$	$[0, \pi/2]$
`Motor 2`	$[0, \pi]$	$[0, \pi]$

Table 3.1: Parameters (lengths (m) and joint limits (rad)) of two-link robots.



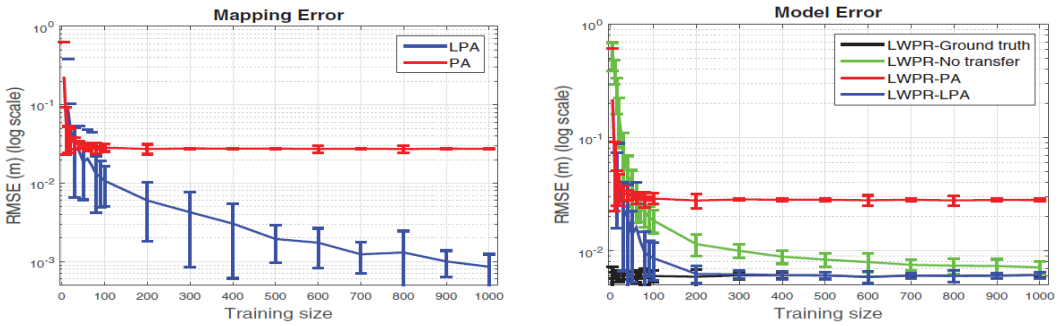
(a) Source workspace.

(b) Target workspace.

Figure 3-5: Error distributions. The scale of the color bar is meters and for visualization purposes, errors above 1 cm are capped at 1 cm.

while keeping the links of the source robot fixed. We find correspondences in the joint space by pairing together points from both robots that correspond to the same joint space position. For LPA, we cluster the points in the 2 dimensional joint space and $N_{min} = 7$ for the two experiments. Fig. 3-5 shows that the end-effector workspaces of the source robot (Fig. 3-5a) and the target robot (Fig. 3-5b) for experiment 1 are not linearly transformable.

We use the LWPR [146] algorithm to learn forward kinematics of the target robot as the ground-truth model from 10 000 uniformly generated points (RMSE = 0.0061 m). Fig. 3-6 shows the performance comparison of the linear method and our non-linear method for different training data sizes for experiment 1. All results are averaged over 100 runs. This experiment shows that LPA offers a more accurate transfer of data and given enough training data no further learning for the target robot is required to obtain an accurate model. We achieve convergence to the ground-truth model with about 200 data points. Without transfer, this would not have happened



(a) Mapping accuracy.

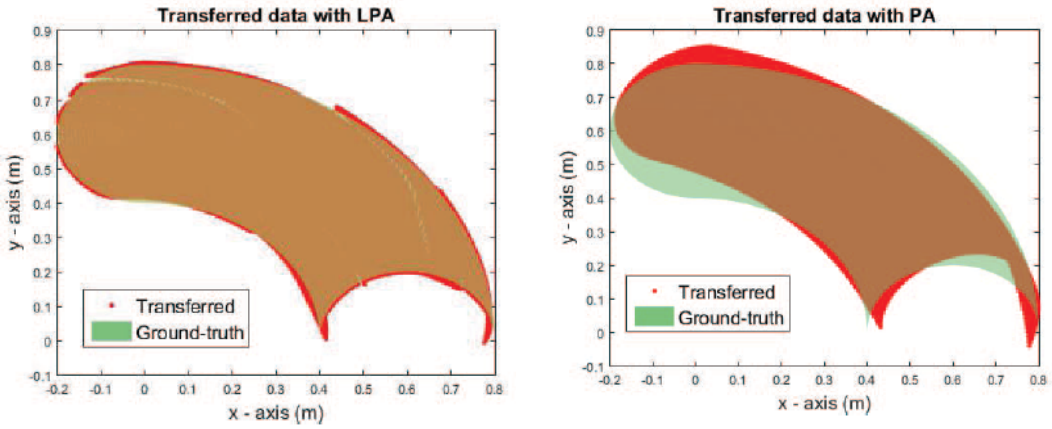
(b) Model error.

Figure 3-6: Mapping and model errors as functions of training data size. (a) The error in the mapping is measured by the RMSE of the transferred points on the ground-truth data. (b) The error in the transferred model is measured by testing 10 000 random samples from the target robot on the different forward kinematics models.

until well over 1000 points. Also, although the linear method converges slightly faster, it does so to a much poorer model.

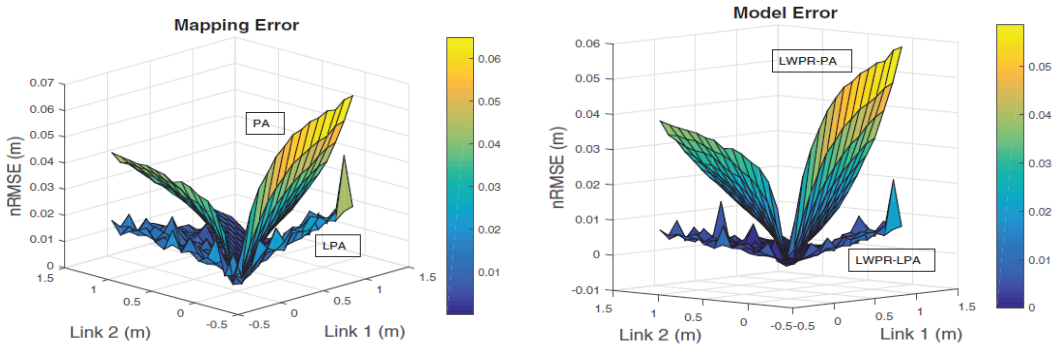
Fig. 3-5 shows the error distribution in the source workspace and the target workspace for an instance with 100 training points. It also shows the Gaussian components fitted to the training data, with the ellipses indicating one standard deviation of each component. Large errors occur in regions where there are either no points or the required transformation is extremely non-linear. In both cases, adding more training data would reduce the mapping errors. Fig. 3-7 shows the transferred data of LPA and PA for a training set of 100 points, superimposed on the ground-truth data, to highlight the resulting structures of the transferred data for both methods.

Fig. 3-8 shows the results for experiment 2. We incremented each link by 0.1 m from 0.1 to 1.5 m and again used a training set of 100 points. The x- and y-axes are the difference between the robots' link 1 and link 2, respectively. We observe that both methods have a zero error along a line in the x-y plane. This occurs when the ratio between the links is the same for both robots, i.e., $\frac{L_1^s}{L_2^s} = \frac{L_1^t}{L_2^t}$ where L_i^s and L_i^t indicate the i-th links of the source robot and target robot, respectively. In this case, the underlying manifold of the target robot is a scaled version of the source robot and therefore the underlying relationship is linear. As this ratio changes the error of the linear mapping increases, while that of LPA stays relatively low. Furthermore,



(a) Local Procrustes Analysis. (b) Procrusts Analysis.

Figure 3-7: Modeled workspace regions using LPA and PA.



(a) Mapping accuracy. (b) Model error.

Figure 3-8: Mapping and model errors as functions of difference in robot links.

LPA still manages to improve the learning of the target model when the difference of the robots is significant and most of the model errors are under 1 cm. Note that errors are high when the ratios are very dissimilar but these can be reduced with more training data.

Chapter 4

Learning Motor Skills from Demonstrations

As the number of robots active in real-world settings continues to grow and their capabilities improving as well, it is important that robots can be adapted to new situations by extending their sets of behaviors or skills. Conventionally, robot skills – also known as policies or behaviors – are developed by hand, where they are hard-coded onto robots by an engineer. However, this requires tedious effort and expertise; and to adapt to new situations, new behaviors need to be hard-coded onto the robots. Recently however, machine learning techniques have been adopted to enable robots to acquire policies from data. This offers opportunities for robots to continuously build new policies as new data arrives, as well as adapt to new situations.

In this chapter we investigate techniques that enable humans to transfer knowledge to robots in order to accelerate policy acquisition. In particular, we aim to initialize the parameters of robot policies from human demonstrations of corresponding skills, in such a way for the robot to quickly acquire the skills when it continues to learn by itself. We propose a data-driven approach based on manifold alignment, that learns to adapt human demonstrations onto the bodies of robots with unknown kinematics. We show that the proposed approach can also be applied to transfer the knowledge gained by one robot from a human to another robot.

In Section 4.1 we introduce this chapter and briefly present an introduction to

robot learning from demonstration. Section 4.2 reviews various methods for transferring human knowledge to robots by means of adapting human demonstrations. We formulate the problem of adapting human demonstrations onto robots and initializing robot policies in Section 4.3 and provide our proposed knowledge transfer solution in Section 4.4. We demonstrate that our solution can reasonably initialize robot policies with experiments in Section 4.5 and conclude the chapter with a discussion of our results in Section 4.6.

4.1 Introduction

Reinforcement learning (RL) is one example of frameworks used in learning new robot skills, where robots learn from their own experiences, through optimization of some reward function provided by a human [131]. In RL, a robot learns one skill at a time. Another framework is developmental learning (DL), where robots are equipped with mechanisms that enable them to autonomously explore their parameterized space of policies/skills to solve a corresponding parameterized space of goals/tasks [6]. In contrast to RL, DL seeks to equip robots with mechanisms for lifelong learning of multiple skills in a developmental manner similar to that of human infants [46]. However, in both frameworks the robot learns from scratch and must physically interact with its environment over many trials/episodes to generate data to update the parameters of its policies. This process is challenging for manipulators and humanoids with high-dimensional and continuous state and action spaces, as this involves long training times. Furthermore, exploration by a robot in uncontrolled environments (e.g., domestic, hospitals, etc.) could be disastrous (e.g., the robot could break nearby objects or fall down the stairs, etc.).

One way to accelerate the learning process, and simultaneously ensure safe robot exploration, is to transfer knowledge to the robot from human trainers. This transfer of knowledge can take many forms, including initializing the parameters of the robot policies with human demonstrations [126, 106, 135, 90, 16], letting the human trainer interact with the robot while it explores the environment and providing it with guid-

ance, in terms of actual trajectories the robot must follow [92] or feedback about its performance [17, 52, 77]. Our work falls under techniques that make use of human demonstrations to initialize the policy of the robot.

In most cases human demonstrations are generated by means of teleoperation, where the robot is operated through the use of a joystick to perform the desired task [126], or through kinesthetic guiding, where a human physically guides the robot through the task [92]. In these techniques, sensors that record the demonstration are typically placed on the robot, so the data sets collected can be directly used by the robot learner, thereby bypassing the correspondence issues [91] – issues that arise due to the differences in the kinematics of the teacher and robot learner, which prevent the collected data sets from being directly used by the robot learner. However, these strategies are limited to robots with fewer DoFs, due to the difficulty of a human controlling each DoF in order to produce a coordinated overall behavior.

An alternative approach, which we adopt in this work, gathers human demonstrations using camera systems, such as motion capture systems. This approach is natural for humans as the human teacher is allowed to perform the task as best as they can, without any obstruction from the robot learner. Furthermore, in this approach, the human trainer can be replaced by an *experienced* robot teacher, allowing for knowledge transfer across robots. However, techniques using this approach must directly confront the aforementioned correspondence issues. Such techniques have been studied under the framework of Learning from Demonstration (LfD), also referred to as Imitation Learning, Programming by Demonstration and Learning by Demonstration.

In LfD, a policy is learned from example data sets provided by a demonstrator. The demonstrator acts as a teacher, either in the form of a human or another robot, performing desired behaviors for the robot to learn. Without loss of generality we will refer to the teacher as a human, as is illustrated in Fig. 4-1. LfD algorithms utilize the provided data sets to derive policies that attempt to reproduce and generalize the desired demonstrated behaviors on the robot [3].

Within LfD, policy acquisition can be facilitated by ordinary users (i.e. non-

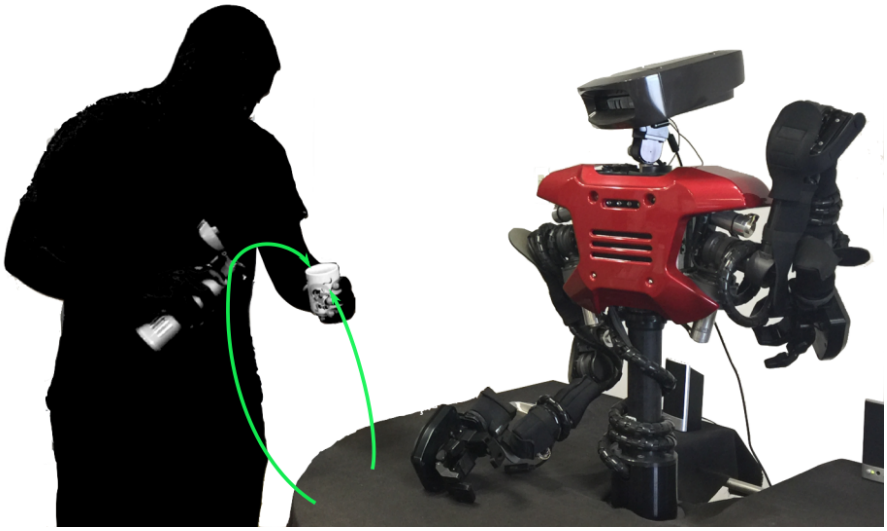


Figure 4-1: Example of goal-directed imitation learning from human demonstrations. A robot learner uses a Kinect sensor to observe a human teacher demonstrate a task of pouring into a cup. The robot learner must be able to reproduce the general task of pouring into the cup, rather than merely mimicking the human posture. The green curves represent trajectories of the objects during teacher demonstration.

robotics-expert users), because its formulations do not typically require domain expertise of the robots and tasks. Furthermore, demonstration is intuitive for humans as they already use it to teach other humans, making it natural to demonstrate tasks to robots. The LfD learning problem can be broadly segmented into two phases: how to gather demonstrations and how to derive policies [3]. Gathering demonstrations is the process of building a data set of examples, which ranges from the selection of sensors for collecting the data – which controls the type of data collected – to the type of demonstration technique to use. We briefly discussed various demonstration techniques and their pros and cons above.

Deriving a policy generally involves encoding the provided data set of examples by learning a model, which can be used to later reproduce the demonstrated task on the robot and generalize to different contexts. Data-driven approaches have received considerable attention recently. Statistical modeling [18, 19] and dynamical systems [61, 59] are amongst the most popular approaches. In particular, dynamic movement primitives (DMPs) are widely used due to their flexibility and stability [59, 143], and have been used to parameterize RL policies [106, 68, 129].

This work focuses on how to adapt demonstrations gathered using cameras, such that they are useful to the robot learner, and to demonstrate that policies can be derived from the adapted demonstrations. The derived policy can serve as an initialization to accelerate a subsequent optimization step of the policy parameters in an agent learning by interacting with its environment – as in RL and DL. Several approaches have been proposed to deal with correspondence issues that arise in this setting, including optimization approaches based on a kinematic model of the robot learner [31, 141, 79], and data-driven approaches that attempt to model a mapping from the teacher space to the robot space [124, 128, 35]. Techniques that rely on a kinematic model of the robot learner are among the most widely used, and some general methods have been proposed [141, 79]. However, in some cases an accurate kinematic model of the robot learner may not be available, limiting the applicability of these methods.

This may be the case, for example, when the specifications required for modeling the robot are not released by the manufacturer [80], and so need to be measured by hand, leading to an inaccurate kinematic model, requiring further calibration. Another example is in dealing with robots whose bodies change over time, potentially as a result of modification, repair, or material damage [130]; or when dealing with biologically-inspired robots, with realistic skeletons and series-elastic, compliant actuators, such as Coman [142] and Meka[1]; and tendon-driven joints, such as the iCub [88]. In these cases, data-driven approaches based on machine learning techniques offer an alternative.

We propose a data-driven approach based on manifold alignment that is suitable in such cases where knowledge of the kinematic model of the robot learner is not available. We also demonstrate and analyze the possibility of transferring skills that have been acquired by one robot from human demonstrations, to another robot. Next we provide an overview of related work in human motion adaptation to kinematically different embodiments.

[1]https://github.com/ahoarau/mekabot/wiki/Meka-robot-overview [accessed June 26, 2017].

4.2 Related Work

The idea of projecting a motion from one kinematic embodiment to another is also found in other areas of computer graphics and robotics, where it is generally referred to as kinematic retargeting [141, 132]. It allows the transfer of gestures or behaviors that are defined in one reference frame (the source) to another (the target) [141]. Techniques generally differ based on the type of motions to be adapted and information to be preserved. For example, to adapt dancing motions from a human dancer to a humanoid, we may be interested in adapting the *joint configurations* of the human dancer, such that the robot can mimic the dance moves, thus preserving postural information. Another example of motions, which is of particular interest to our work, are motions in which we desire the preservation of *goal-directed* characteristics of the movement, where the focus is on achieving some goal, typically with an end-effector of the robot. We refer to this as goal-directed imitation[2].

Example cases where goal-directed imitation is useful include a workshop setting, where it may be desired for multiple robots to perform some tasks demonstrated by a human, such as painting, welding, or pouring fluids as shown in Fig. 4-1. Another example is playing golf, where the robot must swing the golf club such that it strikes the golf ball at a desired location while satisfying some constraints such as via-points and obstacles [79]. For a robot to successfully reproduce such tasks, it must satisfy some constraints in task space, rather than merely mimicking the human movements. In our experiments, we use a goal-directed task of writing letters in the task space, where it is desired that the robot learner reproduce the letters exactly in size and position.

Approaches based on kinematic retargeting for adapting this kind of motion to robots typically assume an accurate kinematic model of the target robot. The general idea is to find an optimal transformation (i.e. locating the task in the robot frame) and adaptation of the demonstration to the target robot, where correspondences

[2]The term goal-directed imitation is also used to mean imitation learning where the teacher's intention is inferred from the demonstrations. Here we use it to describe imitation of whole trajectories in task space.

between the human and the robot are typically known. The demonstrations are generally adapted by maximizing their similarity to the reproductions by the target robot, while satisfying kinematic constraints, such as joint limits and end-effector reach. This includes techniques based on non-linear optimization [38], using Inverse Kinematics (IK) to fit corresponding poses between the human body structure and the robot structure [155], and optimizing a generic weighted cost function whose weights control the similarity of tasks in both task and joint spaces [141]. When correspondences are not known (e.g. adapting to non-anthropomorphic robots), an automatic method that searches for the optimal location and adaptation of the human demonstration, based on the capability of the robot in reproducing it, can be used [79].

When the kinematic data for the target robot is not available, data-driven approaches have been employed as alternatives. Here, a data set of correspondences between the source (human or robot) and target (robot) spaces is collected and a mapping between the spaces is learned, after which this mapping is used to transfer novel points from the source to the target space. Most techniques in the literature transfer human demonstrations in joint space, typically without explicitly addressing goal-directed motions.

Examples include learning a direct mapping from sensor data from a motion capture suit to the position of the robot actuator by training a feed-forward neural network for each DoF [128]; or a two-step mapping process, where the sensor data and robot actuator data are assumed to share a common latent space of lower dimensionality, and the goal is to find mappings from the high-dimensional sensor data to the latent space and then to the high-dimensional robot actuator space. Several techniques have been proposed for learning these mappings, including treating the task as a regression problem and applying Gaussian processes for both mappings [124, 125], using Kernel Canonical Correlation Analysis (KCCA) to map to the latent space and Kernel Regression (KR) to map from the latent space to the robot space [57], and using a mixture of factor analyzers (MFA) combined with a dynamical system for modeling and stable reproduction of trajectories [45].

Another example is using Shared Gaussian Process Latent Variable Models (Shared-GPLVM) to jointly learn a latent representation of skills in a lower-dimensional space [35]. This has shown to be able to use the hyper-parameters of one robot to accelerate learning of the same skills by another kinematically similar robot. This is similar to our work in that knowledge acquired by one robot from demonstrations is transferred to other robots, which reduces the time the human operator spends on training the robots. However, the Shared-GPLVM models as presented assume that the source and target inputs coincide, which is not necessarily the case if the robots do not share the same workspace as the human demonstrator. Furthermore, goal-directed motions were not addressed in this work.

4.3 Problem Statement

Consider a parameterized policy representation $\pi_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta} \in \Theta$ is a vector of policy parameters. The policy encodes a robot skill, i.e. a trajectory that the robot must follow towards some goal in order to perform a task. Here, we consider trajectories in the joint space of the robot $\boldsymbol{q} \in Q$ in terms joint position vector $\boldsymbol{q}$, velocity $\dot{\boldsymbol{q}}$ and acceleration $\ddot{\boldsymbol{q}}$. Learning consists in finding the optimal policy $\pi^{\star}_{\boldsymbol{\theta}}$ by incrementally updating the parameters $\boldsymbol{\theta}$ with data generated by the robot while interacting with its environment, until the parameters $\boldsymbol{\theta}^{\star}$ resulting in the optimal policy are found. Finding the optimal parameters quickly depends on a good initialization and, as previously discussed, learning can take long training times if not initialized properly. Thus, we seek to seed policy learning by initializing it with $\boldsymbol{\theta}^{init} \approx \boldsymbol{\theta}^{\star}$ obtained from adapted human demonstrations.

Assume a given trajectory $\xi^{H}_{1:T} = \{\boldsymbol{q}^{H}, \boldsymbol{x}^{H}\}_{1:T}$ of duration T, provided by a human demonstrator H, consisting at each time step of a d_H dimensional human joint angle vector $\boldsymbol{q}^{H}$ and a human hand (tip, end-effector, etc.) position and orientation (pose) vector $\boldsymbol{x}^{H}$ in task space X^{H}, where d_H is the number of the human joints active when performing a particular task. $\boldsymbol{x}^{H}$ is given in the reference frame of the human demonstrator. A robot R in a different location is to reproduce the demonstrated

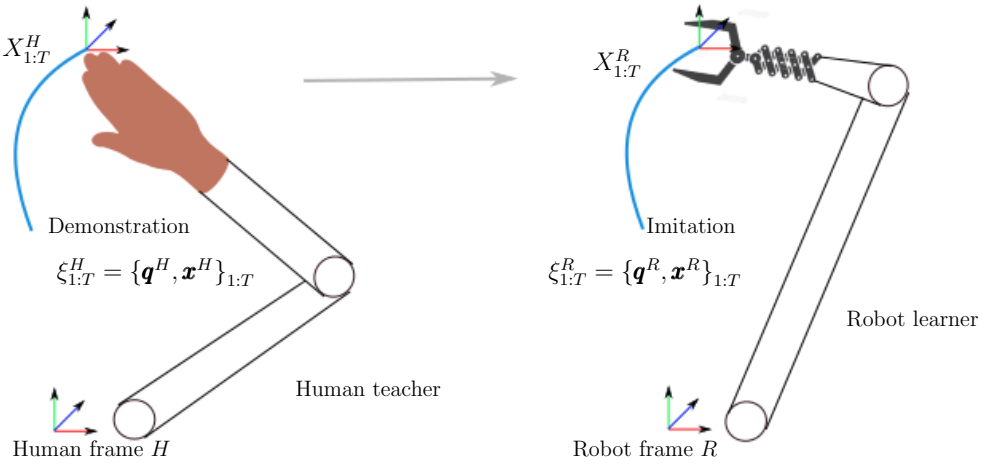


Figure 4-2: Adapting human demonstrations onto robot learner.

trajectory with its own arm w.r.t. to its own reference frame. We seek to adapt $\xi^H_{1:T}$ to $\xi^R_{1:T} = \{\boldsymbol{q}^R, \boldsymbol{x}^R\}_{1:T}$ such that the robot is able to reproduce it, where $\boldsymbol{q}^R$ is a d_R dimensional joint angle vector of the robot, consisting of joints active when the robot is performing the task and $\boldsymbol{x}^R$ is the pose vector of the robot tip in robot Cartesian space X^R. The adapted demonstrations are subsequently used to initialize the parameters of the policy $\pi^{init}_{\boldsymbol{\theta}}$, which can be further fine-tuned by optimizing on the robot. This is illustrated in Fig. 4-2.

We aim to adapt the given human demonstration without assuming a kinematic model of the target robot, in contrast to much of the related work. This can be extended into a multi-robot problem setting (as illustrated by Fig. 4-3 for $n = 2$ robot learners), where the demonstration $\xi^H_{1:T}$ must be adapted to multiple robot trajectories $\xi^{R_1}_{1:T}, \xi^{R_2}_{1:T}, \cdots, \xi^{R_n}_{1:T}$, for n robots. Each trajectory is encoded onto its own skill model $\pi^{init}_{R_1}, \pi^{init}_{R_2}, \cdots, \pi^{init}_{R_n}$ for the corresponding robot. Thus, n different mappings must be learned from the human data to the n robots. In the next section we present our proposed data-driven approach for solving this problem.

4.4 Knowledge Transfer for Motor Skills

Our proposed method employs a data-driven scheme based on manifold alignment, that maps data from the domain of one agent to another. It requires that we provide

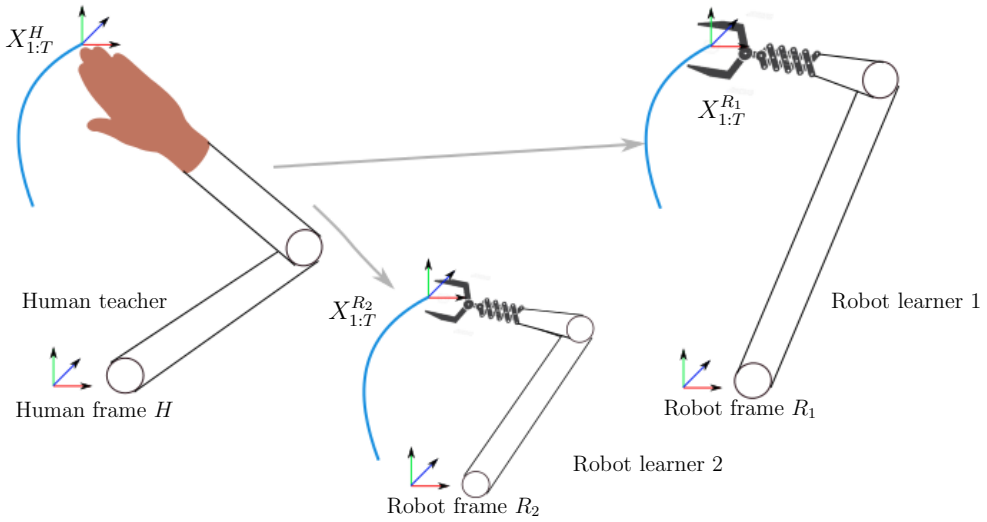


Figure 4-3: Multi-robot problem setting for two robot learners.

corresponding samples from the domains, and a non-linear mapping is learned from these samples. This mapping must generalize to samples from the same domains not seen during training. The domains represent kinematic data generated by a human or robot. By employing a manifold alignment approach, the mapping between domains can be learned from very few samples, compared to the number of samples required to instead learn a kinematic model of the robot, as illustrated in our example in Section 3.6. Below, a high-level overview of our method is presented (Section 4.4.1), followed by the processes of collecting sample correspondences between the domains (Section 4.4.2), learning the mapping (Section 4.4.3), and encoding the mapped trajectories as parametrized skills (Section 4.4.4).

4.4.1 Overview

Figure 4-4 shows the overview of our proposed method. We adopt the approach of projecting human captured data onto a corresponding human skeletal model, because this allows for a unified representation of captured human data from different sensors [5, 47, 139, 82], and also enables our method to be applied in conjunction with different motion capture systems. Given human demonstrations, represented as trajectories of a human skeletal model in joint space and corresponding points in task space, we

aim to map the trajectories onto the body of a robot, such that we obtain joint-space trajectories for the robot corresponding to the human demonstrations. These mapped trajectories are subsequently encoded as parameterized skills for use later in new situations and optimization on the robot.

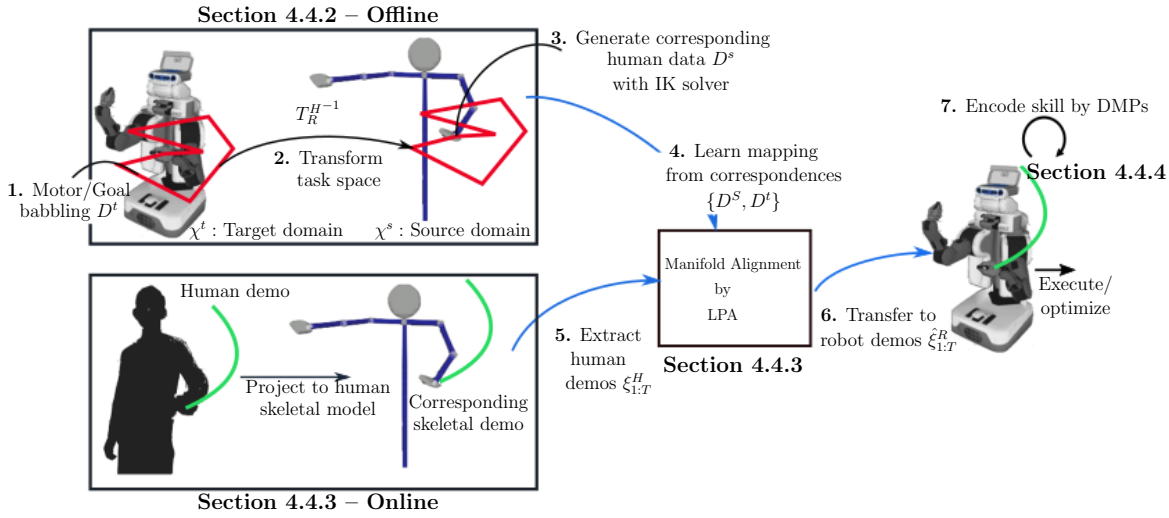
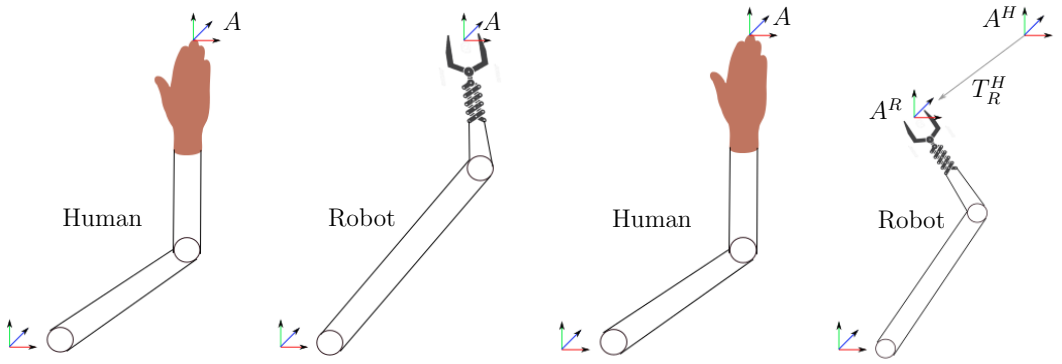


Figure 4-4: Overview of proposed method.

In the first phase of our method, corresponding samples between the human (source) and robot (target) spaces are collected. These samples must be representative of the respective spaces in which the human teacher demonstrates the tasks and the robot learner is expected to perform the tasks. This phase is composed of steps $1-3$ in Fig. 4-4 and is described in Section 4.4.2. Then a non-linear mapping is learned from these samples using LPA in step 4 of Fig. 4-4, as described in Section 4.4.3.

The second phase is composed of steps $5-7$ in Fig. 4-4, where the learned mapping is employed to adapt human joint trajectories onto robot joint trajectories, as described in Section 4.4.3, after which the adapted trajectories are encoded as parametrized skills (see Section 4.4.4). Since the learned mapping is defined globally within the limits of the given spaces, any trajectory that lies in the domain of the human space can be mapped onto the robot space. This consequently allows the transfer of any skills that both the human and the robot are capable of performing within their respective domains.



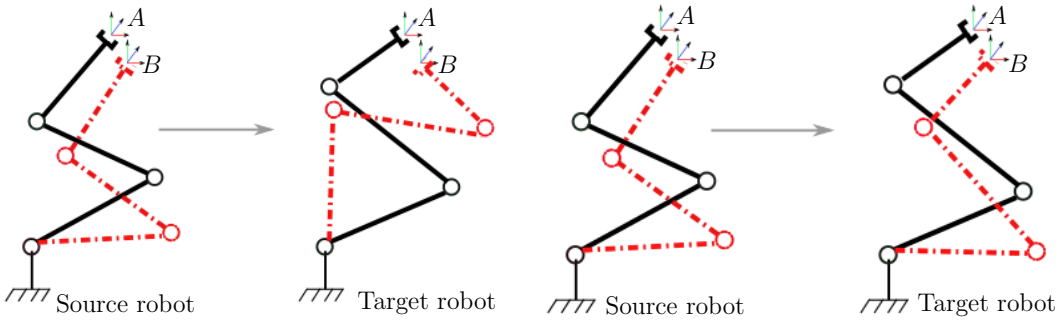
(a) Similar dimensions. (b) Different dimensions.

Figure 4-5: Illustration of corresponding poses. (a) Different arm configurations correspond to each other because they reach the same end-effector point, A. (b) If the robot learner is smaller (or bigger), the task may need to be transformed from the human frame, A^H, to the robot frame, A^R, through the transform T_R^H such that the learner is able to perform it. The same T_R^H is applied to all human data points to transform them to the robot frame.

4.4.2 Correspondence

As described in Section 4.2, goal-directed imitation requires that the robot preserves task-space information. For this reason, we define correspondences as follows: two points in the joint space of the demonstrator and the robot learner are in correspondence if they lead to the *same* end-effector pose w.r.t. to some frame (demonstrator's or learner's frame). This is illustrated in Fig. 4-5a, where both human and robot arms reach the same point A with their end-effectors, with potentially different arm postures. Furthermore, if the robot learner is smaller (or bigger) the task space of the human demonstrator may need to be transformed, by some affine transformation T_R^H, such that there is some overlap with the task space of the robot learner in the vicinity of the tasks. This is illustrated in Fig. 4-5b, where in order for the robot learner to reach point A^H described w.r.t. to the human frame, it must be transformed to A^R, where we consider points A^H and A^R to be the *same* (or in correspondence) across the task spaces.

In kinematic retargeting, correspondences can be easily determined from human demonstrations given a kinematic model of the robot learner, e.g. using an IK solver [155]. Although a kinematic model of the robot learner is assumed not available in



(a) Inconsistent IK solutions for neighboring points A and B due to redundancies.

(b) Consistent IK solutions for neighboring points A and B with redundancy resolution.

Figure 4-6: Illustration of inconsistent IK solutions due to arm redundancies for 3-DoF planar robots in a 2D task space. The black solid and red dashed lines correspond to similar tasks for the same robot. (a) Target data set will have inconsistent neighbors due to redundancies. (b) With redundancy resolution, target data set will have consistent neighbors.

our case, it is available for the human teacher, since the demonstrations are projected onto a human skeletal model. Thus, given some robot data transformed into the task space of the human model, corresponding human joints can be collected by using an IK solver of the human model.

To collect a data set of correspondences, we propose generating N random robot pose data, which can generally be accomplished by employing autonomous robot exploration strategies from DL, such as motor babbling or goal babbling [114, 89] – step 1 in Fig. 4-4. This data consists of robot joint angles $\boldsymbol{q}_i^R$ and their corresponding task space points $\boldsymbol{x}_i^R$, where $i = 1, 2, \cdots, N$. Then we use an IK solver to generate corresponding human joint angles $\boldsymbol{q}_i^H$, from their corresponding transformed task space points $\boldsymbol{x}_i^H$ – steps 2-3 in Fig. 4-4. The affine transformation T_R^H for transforming the task spaces X^H and X^R, is determined manually in our experiments, by placing the task w.r.t. to the robot frame such that the overlap between its task space and the human task space is maximized around the task. However, any method that performs this transformation automatically without a kinematic model of the robot can be employed.

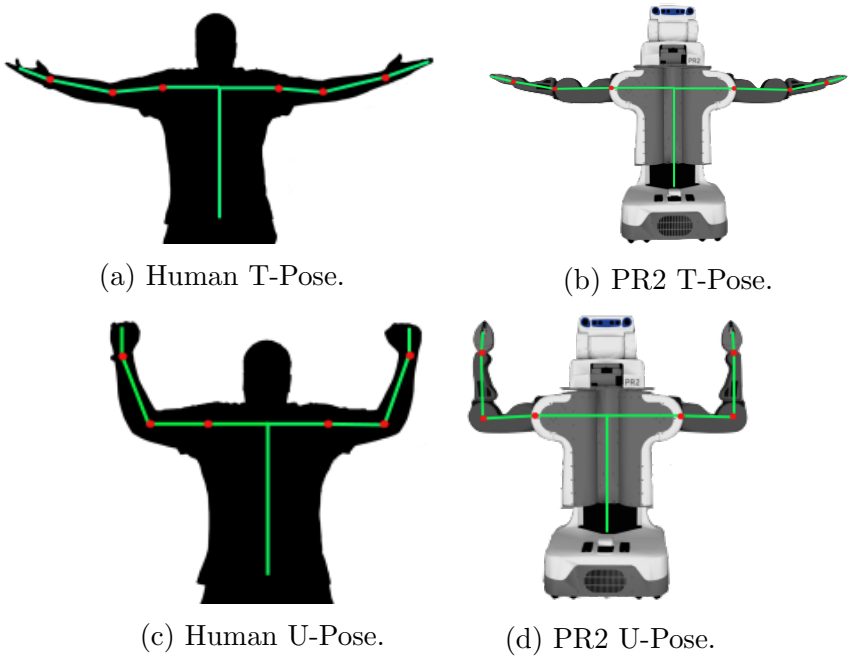
In certain cases both the human arm and corresponding robot joints may be

redundant for a given task. In such cases, multiple joint configurations (corresponding to points potentially lying far from each other in joint space) on both the human and robot can lead to the same task space points, resulting in very different joint configurations for similar end-effector poses – meaning there may be inconsistencies in our samples. This is illustrated in Fig. 4-6a, where nearby configurations of the source robot (black solid and red dashed) lead to nearby points A and B, but can correspond to target robot joint configurations that are far from each other. To avoid this issue we also consider posture *similarity* in joint space: for each human and robot pair, we identify robot joints that correspond to each other for a particular task and determine their pose *similarity*, where heuristics such as the T-Pose [69] or normalized pose [141] can be used to determine pose similarity.

In this work we use a sequence of pre-defined poses, such as T-Pose and U-Pose shown in Fig. 4-7, to easily visualize which joints between the human and the robot correspond to each other, their direction of rotation and how they can map to each other. Given this mapping for all joints involved in solving a task, we can resolve redundancies by initializing the IK solver of the human model with the robot joints that lead to the corresponding point in the robot task space (e.g., A^H and A^R in Fig. 4-5b), by mapping the robot joints onto the joint space of the human model. The resulting mapped values are then used to initialize the IK solver. Note that merely mapping the robot joints does not provide us with human joints that correspond to the *same* point in the task space, but it provides us with joints that are closer to the corresponding joints around the mapped joints, which can be used to initialize IK. This provides us with a consistent set of corresponding samples for a given joint relationship between the human model and the robot, as shown in Fig. 4-6b.

4.4.3 Learning the Mapping

Once a sample of corresponding points has been collected using the method presented in the previous section, we can use it to learn a mapping between two domains – step 4 in Fig. 4-4. To this end, we employ Local Procrustes Analysis. Given samples of correspondences from each domain, $D^s = \{\boldsymbol{q}_i^s, \boldsymbol{x}_i^s\}_{i=1}^N$ and $D^t = \{\boldsymbol{q}_i^t, \boldsymbol{x}_i^t\}_{i=1}^N$, where N is



(a) Human T-Pose. (b) PR2 T-Pose.

(c) Human U-Pose. (d) PR2 U-Pose.

Figure 4-7: Illustration of the T-Pose and U-Pose with the PR2 robot.

the sample size, the objective is to learn a mapping function, $f : \chi^s \mapsto \chi^t$, that maps data points from the source domain to the target domain, through which knowledge can be shared between the domains. We input training data D^s and D^t and latent space dimensionality d to Alg. 3 to learn an LPA model, and obtain LPA parameters $\{\Pi, \Phi\}$.

For a given source trajectory, say a human demonstration $\xi^H_{1:T} = \{\boldsymbol{q}^H, \boldsymbol{x}^H\}_{1:T}$ already projected onto a corresponding human skeleton model, for example using the Master Motor Map (MMM) framework [139], transfer with LPA is performed for each point individually along the trajectory, using Alg. 4, to obtain estimated target robot trajectories $\hat{\xi}^R_{1:T} = \{\hat{\boldsymbol{q}}^R, \hat{\boldsymbol{x}}^R\}_{1:T}$. Then each transferred trajectory can be encoded onto a parametrized skill as discussed below in Section 4.4.4.

4.4.4 Skill Encoding

We use Dynamic Movement Primitives to encode joint space trajectories as parameterized policies $\pi_{\boldsymbol{\theta}}$ and demonstrate that we can recover useful skills from human demonstrations adapted using our method. A DMP is specified by a set of nonlinear

differential equations with well-defined attractor dynamics [59]. For a single DoF trajectory q, the DMP is defined as follows:

$$\tau \dot{z} = \alpha_z(\beta_z(q_g - q) - z) + g(x), \tag{4.1}$$

$$\tau \dot{q} = z, \tag{4.2}$$

$$\tau \dot{x} = -\alpha_x x, \tag{4.3}$$

where x is the phase variable, z is the auxiliary variable and q_g is the desired goal of the movement. Parameters α_z, β_z, α_x and τ define the behaviour of this second order system. If the parameters are selected as $\tau > 0$, $\alpha_z = 4\beta_z > 0$ and $\alpha_x > 0$, then the dynamic system has a unique point attractor at $q = q_g$, $z = 0$. Given the initial condition $x(0) = 1$, Eq. 4.3 is solved analytically by $x(t) = \exp(-\alpha_x t/\tau)$. However, to implement different modulations of the DMP such as phase stopping [143], it is better to keep Eq. 4.3 as a differential equation.

The forcing term $g(x)$ is defined as a linear combination of radial basis functions, which enable the robot to follow any smooth point-to-point trajectory from the beginning of the movement q_0 to the end configuration q_g:

$$g(x) = \frac{\sum w_i \Psi_i(x)_{i=1}^N}{\sum \Psi_i(x)_{i=1}^N} x, \tag{4.4}$$

$$\Psi_i(x) = exp(-h_i(x - c_i)^2). \tag{4.5}$$

Here c_i are the centers of the radial basis functions distributed along the trajectory and $h_i > 0$. For robots with more than one DoF, each degree is represented by Eq. 4.1 - 4.2 with different w_i and q_g, but with a common phase variable x and time constant τ as specified in Eq. 4.3. To approximate any smooth trajectory with a DMP, we need to estimate the weights w_i, time constant τ, and the goal configuration q_g. τ is usually set to the duration of the movement, q_g to the final configuration on the

trajectory, while w_i are estimated from the training data (sampled positions, velocities and accelerations) using regression techniques. See [59] for more details.

The training data here is the estimated target robot trajectory $\{\hat{\boldsymbol{q}}^R\}_{1:T}$ in joint space, which is transferred from human demonstrations by LPA. We compute its first and second derivatives to obtain velocities and accelerations.

4.4.5 Multi-robot Transfer

A straightforward extension of our method to multi-robot systems involves applying the method separately for each robot to learn from the same human demonstrations. This approach, as illustrated in Fig. 4-3, assumes that all robots are learning at roughly the same time, in parallel from the same human demonstrations. This requires learning an LPA model for each teacher-learner pair.

Teaching multi-robot systems is attractive for complex tasks that require collaboration of multiple robots. Such tasks can be solved more easily by combining the unique capabilities of each robot, or faster by extending the area of coverage and range of operation [24, 55]. There may be a case where only one robot is available to learn from a human demonstrator, and after some period of time a new robot becomes available to learn the same skills. This may be the case where a new robot is delivered to a factory with existing robots that have previously learned these skills or a *skilled* robot is shipped to a different location with other robots that must learn the same skills.

In these settings, if the existing robot has *mastered* the skills and a human teacher is not available to teach the new robot, it may be beneficial to transfer knowledge from the existing robot to the new one. In this approach, human demonstrations are transferred to the new robot via the existing robot. The existing robot can master the skills by optimizing the parameters of the corresponding policies through trial-and-error while attempting to perform the skills.

To apply our method in this setting, we learn a mapping from a human teacher to the robot that will then act as a teacher to other robot learners. Then we learn a mapping between the robot teacher and the robot learners using the same procedure

Link	Human	PR2	Meka*
Upper arm	250	400	279
Forearm	250	321	322

Table 4.1: Parameters (lengths) of human and robot models in mm. *Estimated from URDF model.

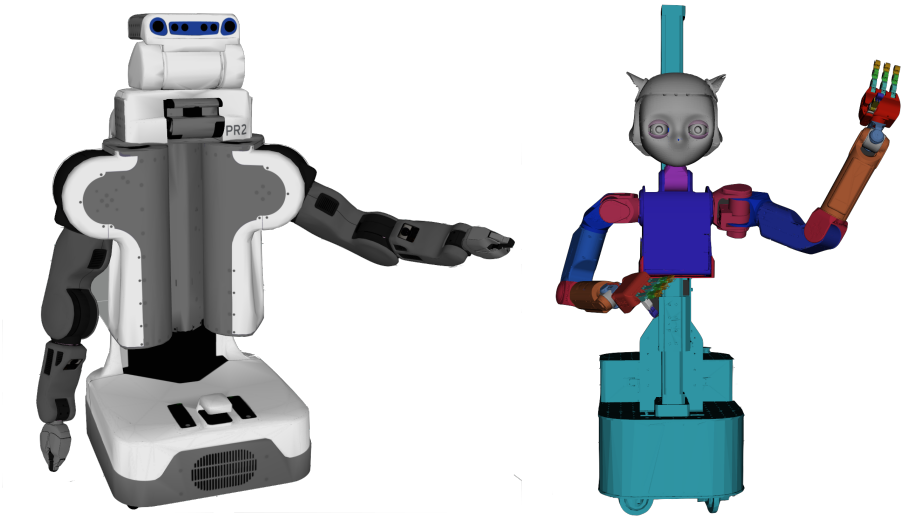
described in our method (Alg. 3), without the step of projecting the trajectories onto a skeletal model. We assume that the robot teacher has successfully learned its kinematic models, or that the models are well understood analytically. This assumption is reasonable, since a robot deployed in some environment for an extended period of time would generate some data from which to learn its kinematics models. Examples of this setting can be found in developmental robotics and lifelong learning.

Under this assumption we can replace the human teacher with a robotic teacher and apply the same method to transfer knowledge between robots. We explore and analyze knowledge transfer between robots in this context experimentally in Section 4.5.3.

4.5 Experiments

To evaluate our transfer method, we designed experiments in simulation to demonstrate and transfer trajectories from a 7-DoF arm of a human model, to two humanoid robots, each with 7-DoF arms, namely the Willow Garage PR2[3] and Meka M1[4], shown in Fig. 4-8. Table 4.1 shows the parameters (lengths) of their arms as well as those of the human model. We demonstrated goal-directed tasks of writing letters, using the common 2D handwriting movement data set[5]. The letters are distributed, scaled and rotated such that they span the 3D task space, as shown in Fig. 4-9. We also analyzed the transfer of the demonstrated tasks from one robot to another robot, to evaluate the possibility of knowledge transfer between robots.

[3]http://www.willowgarage.com/pages/pr2/overview [accessed June 26, 2017]
[4]https://github.com/ahoarau/mekabot [accessed June 26, 2017]
[5]https://gitlab.idiap.ch/rli/pbdlib-matlab/tree/master/data/2Dletters [accessed June 26, 2017]



(a) PR2. (b) Meka.

Figure 4-8: Robots used in our experiments.

The data set contains handwritten letters from A to Z, with several demonstrations per letter. In our experiments we only used 5 demonstrations for each letter in Fig. 4-9. To model human demonstrations, we used an IK solver on the left arm of the human kinematic model to trace the letters and obtained the tasks in joint space; however, these demonstrations could have been recorded by any motion capture system and adapted onto the human model, using for example, the Master Motor Map (MMM) framework [139]. We then used our method to adapt these trajectories onto the left arms of the robots and encoded them using DMPs.

To evaluate our method we executed reproductions of the tasks on the robots, by computing forward kinematics on the simulated robots, to obtain the adapted trajectories in task space. We then also used an IK solver on the robots to trace the letters and obtained ground-truth joint space trajectories of the tasks, encoded them using DMPs and executed the reproductions on the robots to obtain their corresponding reproductions in task space. As a measure of performance, we computed the error in reproducing the letters in task space, using our method and the IK-based method. Our hypothesis is that if the reproduction of the transferred skills is close to the ground truth transfer would serve as a good initialization of policy parameters.

There are multiple ways in which a 7-DoF arm can write the letters in the 3D task

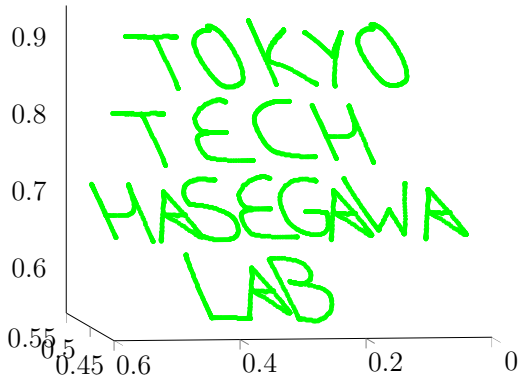


Figure 4-9: Task of writing letters in the task space of the human model.

Table 4.2: Joints mapping between the human model and the robots. J_i is the i-th joint of the human model.

Joint #	Human	PR2	Meka
1	J1	J2	$\pi/2$-J1
2	J2	J1	-J2
3	J3	J3+$\pi/2$	J3+$\pi/2$
4	J4	J4	J4

space. However, we found that this task can also be completed with only the first 4 of the 7 DoFs – resulting in the tasks lying in a 4-dimensional joint space of the human and the robot arms. The domains χ^s and χ^t, data sets D^s and D^t, and trajectories ξ^s and ξ^t are therefore 7-dimensional – 4 in joint space and 3 in task space. As described in Section 4.4.2, we identified correspondences between the 4 DoFs of the human and the robot arms, as shown in Table 4.2. For example, in the first row, the first joint of the PR2 corresponds to the second joint of the human model; and the first joint of the Meka corresponds to the first human joint and the conversion to the Meka joint is $\pi/2$ minus the human joint.

To generate training data for learning an LPA model for each robot, we simulated motor babbling by randomly sampling the 4-dimensional joint space of the robot and applying forward kinematics to obtain corresponding end-effector positions; and for each robot data point we transformed it into the frame of the human demonstrator using $T_R^{H^{-1}}$ and used an IK solver to collect the corresponding human data point, where the IK was initialized by mapping the joints of the robot onto the human

joint space using the inverse of the mappings in Table 4.2. The rest of the joints (last three joints) were kept at constant values. The rotational components of the transformation matrix T_R^H for both robots are set to the Identity matrix, and the translational components are set to $[0.2, 0.0, -0.72]$ for the PR2 and $[-0.1, 0.0, 0.48]$ for the Meka. The values are chosen so as to locate the tasks in the robots' frames w.r.t. to the human frame.

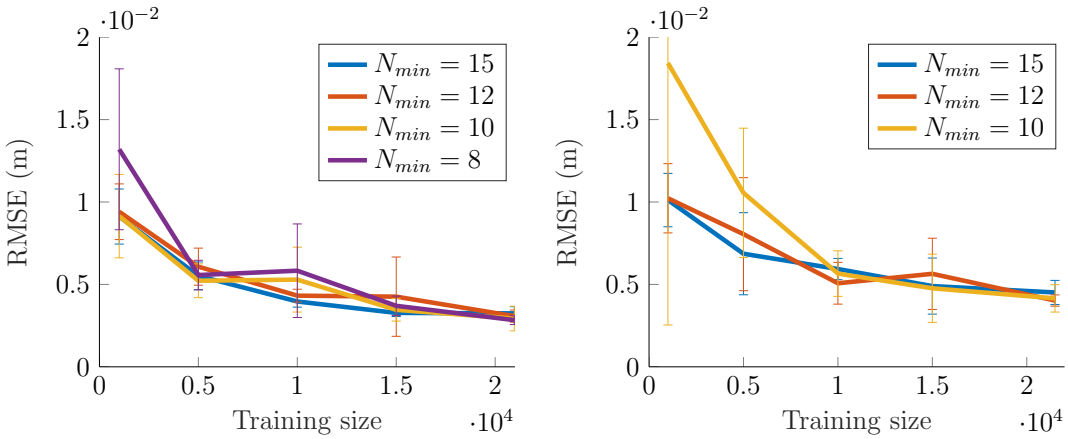
The 5 demonstrations of the letters in Fig. 4-9 were used as test data, and the data set was collected by using an IK solver on the human model to trace the letters. The ground-truth data for the robots was also collected in a similar manner. We start by evaluating the accuracy of mapping the trajectories from the human model onto robots in Section 4.5.1, followed by evaluating the encoding of the mapped trajectories using DMPs in Section 4.5.2, and finally we analyze the transfer of acquired knowledge by one robot to another robot in Section 4.5.3.

4.5.1 Mapping Accuracy

In this section we analyze the mapping of 5 demonstrations for each letter in Fig. 4-9. To train LPA (see Alg. 3), we set C_{min} to 0.005, encouraging narrow clusters, and we experimented with several values of N_{min}. Encouraging narrow clusters runs a risk of overfitting the mapping but this can be controlled by choosing a large value of N_{min}. Figure 4-10 shows the accuracy of mapping the test data to both robots for increasing training sample size and different values of N_{min}. The results were averaged over 5 runs.

We observe that the mapping improves with more data and that it degrades when the value of N_{min} decreases, from 15 to 8 for the PR2 and 15 to 10 for the Meka. Furthermore, the mapping errors are on average less than 0.01 m. Smaller values of N_{min} cause overfitting of the mapping for insufficient training data, indicated by large errors for small training data size. $N_{min} = 8$ overfits the mapping for the PR2 and $N_{min} = 10$ overfits for the Meka.

Figure 4-11 shows samples of quantitative results of the transferred trajectories in green, for $N_{min} = 15$ and sample size of 15k for the PR2 and $N_{min} = 12$ and sample



(a) Accuracy for the PR2.

(b) Accuracy for the Meka.

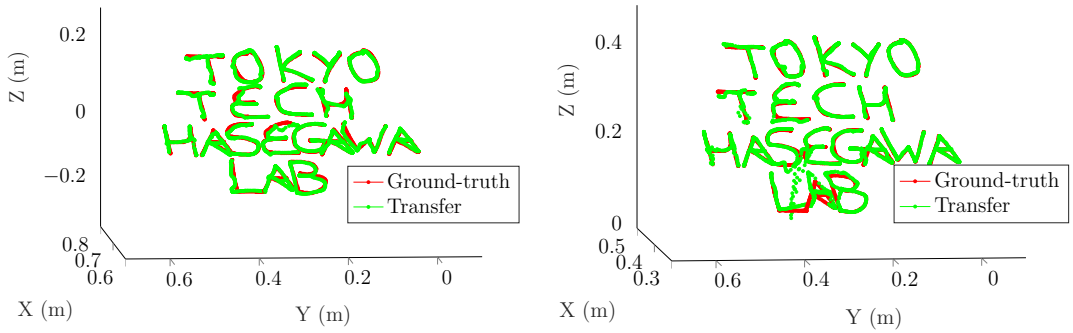
Figure 4-10: Mapping accuracy for the PR2 and Meka.

size of 21k for the Meka, overlaid on the red ground-truth trajectories. Most of the letters look to be mapped accurately; except for A in LAB, for the Meka. In the next section we will see that encoding a skill with DMPs can recover a smooth generalized task even under some mapping errors for the letter A.

The results indicate that mapping to the PR2 is more accurate and requires less data than mapping to the Meka. This is most likely due to the relative differences of the robots w.r.t. to the human model. We observed in our illustrative example in Section 3.6 that the complexity of the mapping required between manifolds of agents increases with the difference between the ratios of their arm links. The ratio of the first link to the second one of the human model is 1, and it is 1.25 for the PR2 and 0.87 for the Meka. Here the mapping is less accurate when the ratio of the robot links is less than that of the human, compared to when it is larger.

4.5.2 Skill Transfer and Encoding

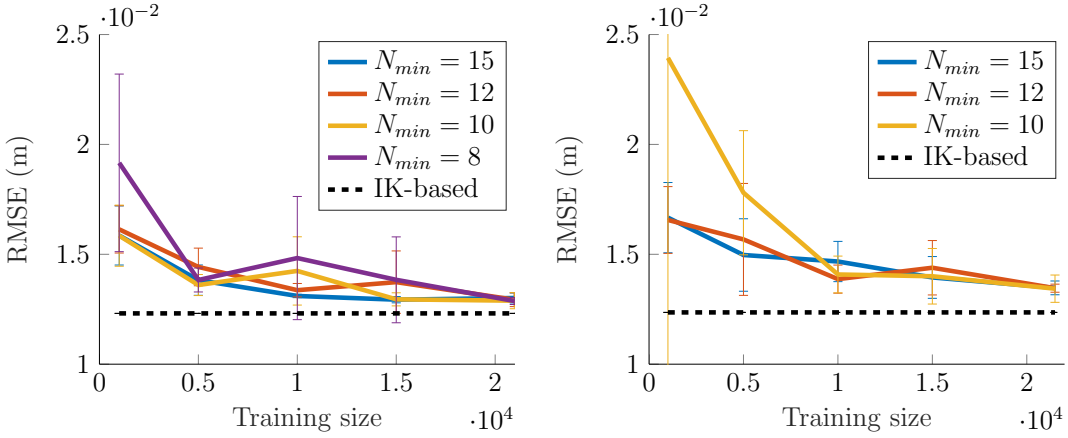
In this section we analyze the encoding of the mapped trajectories as parametrized skills using DMPs. We learned a DMP model from the 5 transferred demonstrations for each letter. We then attempted to reproduce the first demonstration of each letter from the DMP models and compared these reproductions against their corresponding original demonstrations. We repeated this for the IK-based method and averaged the



(a) Transferred trajectories for the PR2. (b) Transferred trajectories for the Meka.

Figure 4-11: Sample transferred raw trajectories for the PR2 and Meka.

results over 5 runs.

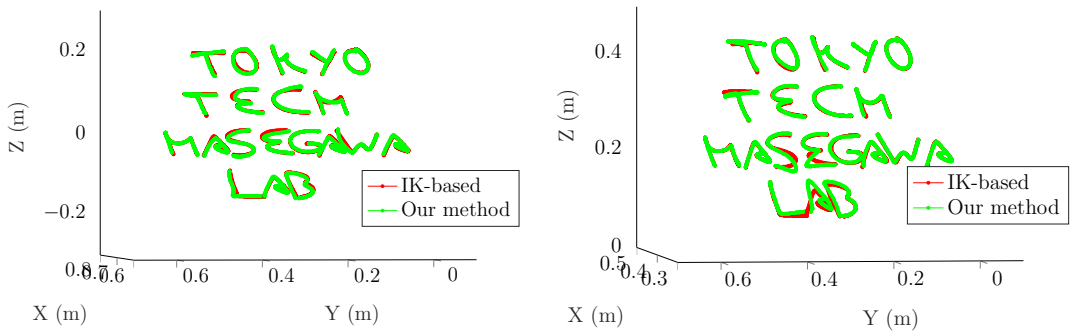


(a) Accuracy for the PR2. (b) Accuracy for the Meka.

Figure 4-12: Encoding accuracy for the PR2 and Meka. The black dashed line corresponds to the results of the baseline method.

Figure 4-12 shows the accuracy in reproducing the letters for different values of N_{min} and increasing training sample size (similar values as the previous experiment). The results of the IK method are shown in black, dashed lines. We note that the effectiveness of encoding a skill is correlated with the performance of mapping the trajectories, as one would expect – inaccurate adaptation of the trajectories would lead to poor encoding and reproduction of a skill by the robot.

The reproductions of the letters using our method are comparable with the reproductions of the IK-based method, which makes use of kinematic models of the robots. Some letters with complex shapes, such as H, K, A and B, could not be reproduced



(a) Imitation for the PR2.

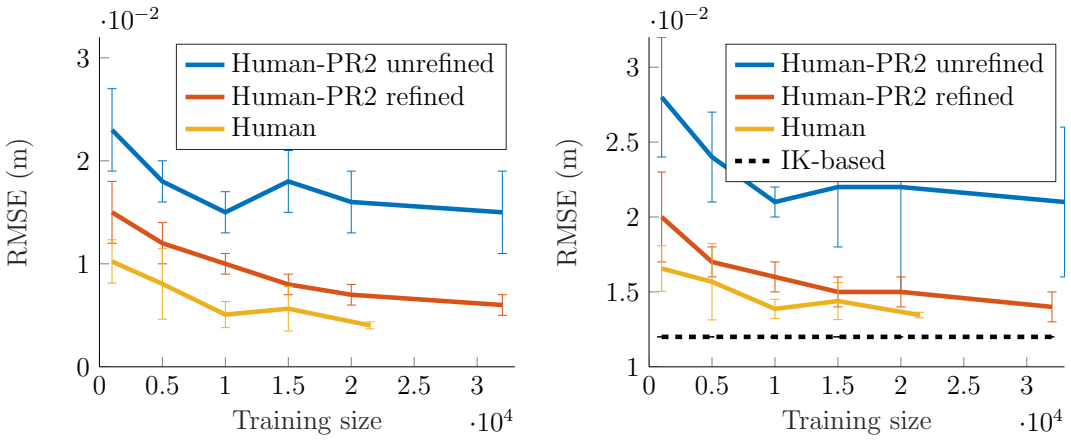
(b) Imitation for the Meka.

Figure 4-13: Sample task imitation for the PR2 and Meka.

exactly by DMP, accounting for much of the errors in Fig. 4-12, for both our method and the IK method. This is most likely because DMP aims to reproduce a generalized skill that can be adapted to new contexts, rather than an exact skill as demonstrated. Nevertheless, all the letters reproduced by the robots were recognizable, as shown in Fig. 4-13. Compared to the raw transferred letters in Fig. 4-11 that are jagged in some areas, the DMP was able to reproduce smooth generalized letters. This is particularly clear for A in LAB in Fig. 4-13. The results presented here demonstrate that we are able to recover the skills from the adapted demonstrations.

4.5.3 Knowledge Transfer between Robots

In the previous experiments we demonstrated the transfer of skills from a human teacher to two robot learners. The same trajectories were demonstrated once and adapted to the two robots. In this experiment we analyze the transfer of knowledge between robots, where the PR2 is the teacher and the Meka is the learner. We assume that the teacher robot either has successfully learned its kinematics models, or that they can be modeled analytically. We analyze the transfer of knowledge from the existing robot to the new one, in comparison with the direct transfer from a human teacher to the new robot. We follow the same procedure as with the previous experiments and only substitute the human teacher with the robot teacher. Since the PR2 is the teacher, we must locate the demonstrations in the Meka w.r.t. to the PR2 frame using T^{pr2}_{meka}, where we set the translational component of the transformation



(a) Mapping accuracy.

(b) Encoding accuracy.

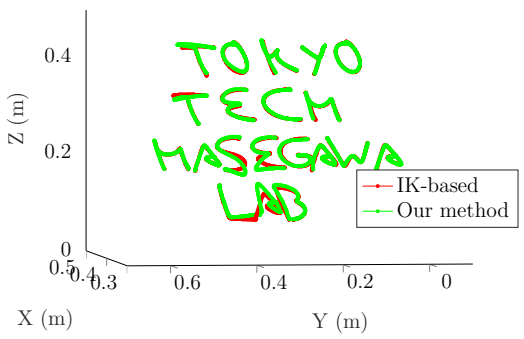
Figure 4-14: Accuracy of transferring tasks of writing letters from the PR2 and human teacher to the Meka.

to $[-0.3, 0.0, 0.23]$ and the rotational component to the Identity matrix[6].

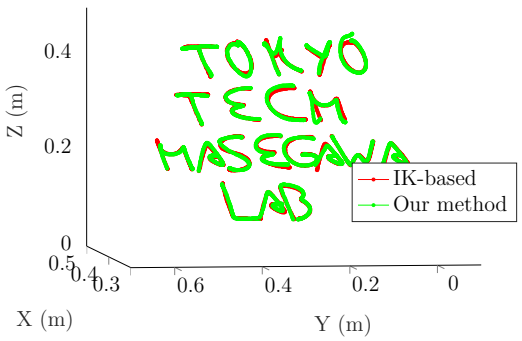
Firstly we transferred skills demonstrated by a human to the PR2 robot and then immediately transferred them to the Meka via the PR2, where the PR2 has not yet mastered the skills. This is equivalent to cascading the mapping from the human domain to the PR2 and the mapping from the PR2 to the Meka. Separately, we also transferred the skills from the PR2 to the Meka, where the PR2 is allowed time to master the skills transferred from the human teacher. The teacher robot can master the skills by attempting to execute them and optimizing the policy parameters using the data generated in a trial-and-error fashion as discussed Section 4.4.5. In this paper, we assume the teacher is capable of this, and thus we model refined robot skills using an IK solver on the robot teacher, the same way we modeled human demonstrations. The former approach is labeled 'Human-PR2 unrefined' and the latter 'Human-PR2 refined'. These two approaches are compared with direct transfer from the human teacher to the Meka.

Figure 4-14 shows the comparison of adapting demonstrations onto the Meka using the three approaches and encoding them using DMPs. Results for transferring directly from human demonstrations are taken directly from the previous sections,

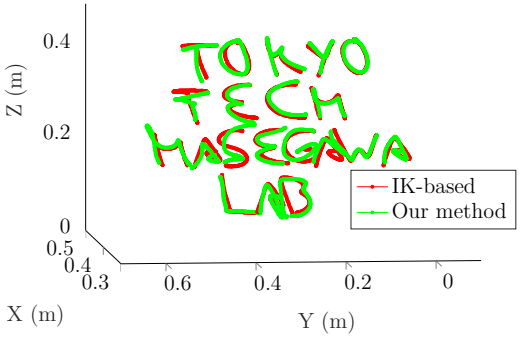
[6]Similar to transfer from human teacher to robot learner, $T^{teacher}_{learner}$ is chosen manually here, such that the learner can perform the tasks in its task space.



(a) Human to the Meka.



(b) PR2 to the Meka refined.



(c) PR2 to the Meka unrefined.

Figure 4-15: Transferred tasks reproduced by the Meka.

for $N_{min} = 12$. We observe that direct transfer from the human teacher is more accurate, followed by transferring refined skills via the PR2. Given more training data, transferring refined skills via the PR2 becomes as accurate as transferring directly from the human teacher. This could be due to the mapping from the PR2 to the Meka being more complex compared to the mapping from the human teacher, since the ratio of the Meka links is much smaller than the ratio of the PR2 links, compared to the ratio of the human links. Transferring unrefined skills via the PR2 is even less accurate because of the accumulation of errors from the human domain.

In Fig. 4-15, we show some quantitative results of transferring to the Meka, using the three approaches. The generalized letters reproduced are recognizable and comparable to the IK method. Table 4.3 summarizes the results, where we show the minimum errors achieved by each approach. Direct transfer from the human teacher and transfer of refined skills from the PR2 are within the standard deviations of each other, and therefore are not statistically significantly different from each other. On

Table 4.3: Comparison of errors for transferring to the Meka, from human and the PR2. The numbers in brackets are standard deviations.

Source	Human	Human-PR2 refined	Human-PR2 unrefined
`mapping (m)`	0.0045	0.0062	0.0152
	(0.0008)	(0.0041)	(0.0035)
`encoding (m)`	0.0140	0.0138	0.0192
	(0.0013)	(0.0005)	(0.0012)

the other hand, transferring unrefined skills from the PR2 is the least accurate and further refinement of the skills by the robot learner would be needed.

4.6 Discussion

The results presented in Section 4.5.1 demonstrated the efficacy of our proposed data-driven approach in adapting human demonstrations onto the arms of two humanoid robots with different kinematic parameters; and the results in Section 4.5.2 confirmed that skills on the robots can be successfully recovered from the adapted demonstrations. In contrast to the standard approach taken by other methods, we showed that generalized forms of the tasks can be reproduced by the robots, while preserving the goal-directed characteristics of the tasks, i.e., preserving the shape and size of the letters, rather than mimicking the posture of the teacher.

Results presented in Section 4.5.3 demonstrated the possibility of human-robot knowledge transfer in a multi-robot setting, where one robot acquired knowledge from a human teacher and then acted as a teacher to a new robot. Due to the accumulation of errors when transferring from a human to the new robot via the existing one, our results show that it is better for the teacher robot to first optimize its skills, or alternatively the new robot can further optimize its skills after transfer, where the transferred knowledge acts as prior knowledge for accelerating the learning process of the new robot.

Our simulation results demonstrated that our approach is capable of human-robot knowledge transfer for robots with unknown kinematics and also knowledge transfer

across robots, and that the transferred data can be used to reasonably initialize the parameters of a policy. As part of our future work, we plan to evaluate the benefit of our approach in accelerating learning of a robotic agent interacting with its environment, by comparing optimization with our initialization and learning from scratch. A possible extension of our work is applying it to transfer human skills to non-anthropomorphic robots, where the challenge is that correspondences with the human body are not obvious.

Chapter 5

Model Learning for Control

Given joint-space trajectories for the robot to follow, obtained either through reinforcement learning techniques[1] accelerated by human demonstrations as discussed in Chapter 4, or through traditional motion planning techniques, a control policy is required to track the trajectories as close as possible. In this chapter we present a model-based control algorithm based on learning the inverse dynamics model, and demonstrate how we can accelerate the learning process by re-using data generated by pre-existing robots.

In Section 5.1 we present an introduction to learning inverse dynamics for control and also provide a brief review of state-of-the-art learning algorithms for this model. Section 5.2 reviews various methods for accelerating learning of inverse dynamics, including techniques based on knowledge transfer. We formulate the problem of learning inverse dynamics in Section 5.3 and provide a solution to accelerating the learning process with knowledge transfer in Section 5.4. We demonstrate the benefit of this solution with experiments in Section 5.5 and end the chapter with a discussion of our results in Section 5.6.

[1]Note that this is not always the case and in some applications learning motor skills with reinforcement learning techniques finds a policy that directly computes robot joint torque.

5.1 Introduction

To control a robot manipulator to follow a specified trajectory, model-based control offers many advantages over traditional PID-based control, including potentially higher tracking accuracy, lower energy consumption and lower feedback gains – which results in more compliant and reactive control [96]. The model is used to predict, for example, the joint torques given the desired trajectory in terms of joint positions, velocities and accelerations. However, the performance of model-based control relies heavily on the accuracy of the models used in capturing the dynamics of the real system under control and its environment. The dynamics model can be developed from first principles in mechanics, based on the Rigid Body Dynamics (RBD) framework [42], resulting in a parametric model, with parameters such as the inertial parameters of link mass, center of mass and moments of inertia, and friction parameters, that must be estimated precisely.

In practice, however, it is difficult to obtain a sufficiently accurate dynamic model for many modern robotic systems based on the parametric RBD model, due to unmodeled non-linearities such as friction, backlash and actuator dynamics. Thus, several assumptions are made to simplify the process, such as rigidity of links or that friction has a simple analytical form, leading to inaccuracies in the model. The inaccurate model can lead to large tracking errors, which must be compensated for using high-gain PID control. As high-gain control would turn the robot into a danger for its environment in reducing compliance, more accurate models are needed.

Learning robot models for control based on regression techniques is typically employed as an alternative in such cases where the physical parameters of the robot are unknown or inaccurate. Unknown non-linearities can be taken into account as the model is estimated directly from measured data, while they are typically neglected by the standard RBD model [95]. Furthermore, as learning-based control techniques are capable of modeling complex systems, modern robots no longer have to be designed to allow modeling to be straightforward but can be designed to suit various demands and environments. As a result, learning approaches to robot modeling have attracted

much interest and have been used successfully in recent years.

In this chapter, our focus is on model-based control for robot manipulators, specifically inverse dynamics control; however, other robot models can be learned, including forward and inverse kinematics [40, 11] (also see Chapter 6), operational space control [105], and forward dynamics [107]. The inverse dynamics model is usually learned using state-of-the-art non-parametric regression techniques. Real-time online learning is preferred over batch learning because i) it allows adaptation to changes in the robot dynamics, load, or actuators; and ii) it is almost impossible to explore entirely the state space of robots with large DoFs in a batch setting.

Real-time online learning of the inverse dynamics model can be broadly broken down into two categories: global methods and local methods. Global methods model a regression function that is defined globally in input space; whereas local methods seek to partition the input space into smaller regions and define a function only valid locally for each region. The final prediction is typically a weighted sum of all local predictions. Examples of global methods include those that make use of the entire available data, such as those based on deep learning methods [108], random features and Ridge Regression [48], and methods that make use of *basis vectors* (or a sparse set) representing the input space [97, 93, 49].

Local methods are inspired by the idea of locally weighted learning (LWL) for control [4] and include techniques that build locally linear models such as Locally Weighted Projection Regression (LWPR) [146], locally non-linear models such as Local Gaussian Processes (LGP) [98] and Local Gaussian Regression (LGR) [84, 85], and Local online Support Vector Regression (LoSVR) [25]. Drifting Gaussian Processes is also a local model, specifically aimed at streaming data [87].

Real-time online learning of the inverse dynamics model using any of the techniques presented above necessitates an interaction of the robot with its physical environment to collect training samples over many trials. This can be challenging for manipulators and humanoids, mainly due to their large and high-dimensional state and action spaces, as a large amount of data must be collected over time – resulting in a time-intensive process. Furthermore, their physical embodiment allows only for

a limited time for collecting training data.

This problem becomes considerably worse in a heterogeneous multi-robot setting, each robot having different structural properties (link dimensions, number of degrees-of-freedom (DoF), etc.), where each robot must go through the same laborious process so as to learn models from scratch. Thus, in this chapter, we aim to accelerate the learning process of a new robot with knowledge transfer, where we re-use data generated by a pre-existing robot as an additional data set for the new robot. By re-using data generated by other robots as a means of knowledge transfer we show that learning for the new robot will be biased towards relevant spaces such that fewer trials of interacting with the environment are needed, thus improving the learning speed.

We propose a scheme for generating training data online from the robots, which we use to learn transfer learning models, and in contrast to previous work in transfer learning for inverse dynamics control, we demonstrate the benefit of knowledge transfer for accelerating online learning of the inverse dynamics model. Prior to presenting our knowledge transfer approach, in the next section we review other techniques that can be used to accelerate learning of the inverse dynamics, including those based on knowledge transfer.

5.2 Related Work

The learning algorithms presented above disregard any prior knowledge about the robot system that may be available, and so begin learning from scratch. One of the techniques for obtaining faster learning speeds that has been proposed recently is marrying the analytical physics-based Rigid Body Dynamics model (if available) with state-of-the-art non-parametric learning models into a semi-parametric model. The benefits of learning such models are faster learning speeds, higher accuracy, and better generalization [94]. The parametric RBD model component acts as prior knowledge and is defined in the entire state space, and the non-parametric component models the non-linearities and adapts to changes online.

Examples include techniques that incorporate the parametric RBD model into the

non-parametric part as a mean function [94, 28], kernel function [94, 116], and those that use first order approximations of the RBD equation to initialize the local models of the LWPR model [34, 32]. Other techniques instead model the inverse dynamics error (or residual), using random features and Recursive Regularized Least Squares (RRLS) [20], or as a constant offset that is continuously adapted via online gradient descent to minimize the tracking error [112, 86].

Another set of approaches for accelerating learning of inverse dynamics, which is of particular interest to our work, is based on the concept of transfer learning [101], where knowledge gained while solving one task (a *source task*) is leveraged to help improve learning a new task (a *target task*). Transfer learning approaches for robotics can be broadly broken down into two categories: *inter-task transfer* and *inter-robot transfer*.

In inter-task transfer, knowledge gained by the same robot while previously performing some tasks is leveraged to speed up learning of a new related task in a different context. This includes the Independent Joint Learning (IJL) method for speeding up and generalizing learning of a new task, possibly in a different state space of the robot [144]. This has been shown to be able to generalize the inverse dynamics model of the Kuka LWR IV+ for load variations [122]. Another approach uses the multi-task learning framework, where Gaussian process priors are employed to share information for learning inverse dynamics for varying loads [152, 21]. In other approaches policies learned in simulation are adapted to the real system, by collecting experience on the real system and training a deep neural network inverse dynamics model [26].

In inter-robot transfer, a data set generated by another robot (a *source robot*) performing a task is used to aid learning of the same task by a new robot (a *target robot*). In general the source and target robots may not be exactly the same (i.e., they may differ in structural properties such as link length and mass, number of links and DoFs, etc), so the source data must be mapped into the domain of the target robot such that it is useful to the target robot. Not a lot of work dealing with this case exists and the majority of those available model this mapping using manifold alignment techniques [148, 149].

A manifold alignment based transfer learning algorithm has been used to demonstrate the possibility of transfer for inverse dynamics between two simulated robot manipulators [8]. An upper-bound on the error of this model was theoretically analyzed and derived with the goal of identifying cases in which transfer is possible and beneficial; it was only validated for linear time-invariant (LTI), single-input, single-output (SISO) systems [111] and was applied to transfer knowledge for linearized models of unicycle robots [110]. A theoretical study was done on SISO systems to show that an optimal map between two different systems is a dynamic map, and an algorithm for identifying the optimal dynamic map was proposed and validated on simulated planar robots and experimentally on two different quadrotor platforms, where the systems were modeled as SISO systems [54].

In some cases it has been shown that the learning speed can be improved by initializing the model with random data generated through a motor babbling process [33, 105]. Here, the robot tracks random joint movements using a PID controller while the parameters of the learner are updated using the data generated. This idea was used to improve learning of inverse dynamics using LWPR [33, 32] and for initializing learning of operational space control [105].

In this work, we investigate the acceleration of learning the inverse dynamics model using inter-robot transfer. Based on our review this approach has not received much attention. The authors of [8] demonstrated the possibility of improving inverse dynamics based on manifold alignment. However, since their aim was to show the soundness of transfer, they did not address how to collect training data from the robots without using analytical models of the robots and so did not demonstrate the benefit of transfer in a realistic scenario. Here, we leverage motor babbling techniques to generate the training data from the robots and demonstrate the benefit of transfer under this training scheme (see Section 5.4.1). The robots are controlled to track random trajectories using PID controllers to generate the training data. Although our experiments are performed in simulation, we do not make use of any analytical models of the robot dynamics to generate training data.

5.3 Problem Statement

The inverse dynamics model of a robot manipulator relates the joint positions $\mathbf{q}$, velocities $\dot{\mathbf{q}}$, and accelerations $\ddot{\mathbf{q}}$ with the corresponding forces and torques $\boldsymbol{\tau}$ required to follow a specified joint-space trajectory; and can be described analytically using the well-known Rigid Body Dynamics formula

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}), \tag{5.1}$$

where $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{g}$ are the inertial, Coriolis, and gravitational terms, respectively [127]. The feed-forward torque $\boldsymbol{\tau}_{ff}$ for the current desired state $\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d$ is predicted using Eq. 5.1, while a feedback torque $\boldsymbol{\tau}_{fb}$, computed using a feedback controller (e.g. a simple PID controller), is used to stabilize the system. Therefore the total torque applied to the robot is $\boldsymbol{\tau}_{tot} = \boldsymbol{\tau}_{ff} + \boldsymbol{\tau}_{fb}$.

Unfortunately, as discussed previously, this formulation is limited when used to control modern robotics systems, due to the difficulty in accurately determining their complex kinematic and dynamic parameters. Thus, learning the model from data generated by a robot, using non-parametric machine learning techniques has emerged as an alternative. Here, the problem is reduced to a standard supervised learning setting

$$\boldsymbol{\tau} = \mathbf{D}(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}) + \epsilon, \tag{5.2}$$

where we seek to learn the dynamic model $\mathbf{D}(\cdot)$ from input-output pairs $\{(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}), \boldsymbol{\tau}\}$ generated by the robot and $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ is the output noise modeled as Gaussian noise with zero mean and variance σ^2. Figure 5-1 shows a learning-based control framework employed in this work.

In an online learning setting, the current torque prediction $\boldsymbol{\tau}_{ff}$, predicted using the current learned model, with the corresponding stabilizing feedback torque $\boldsymbol{\tau}_{fb}$, is applied to the robot, which results in the actual state $\mathbf{q}_a, \dot{\mathbf{q}}_a, \ddot{\mathbf{q}}_a$. The data generated in each time step is immediately used to update the parameters of the model. In

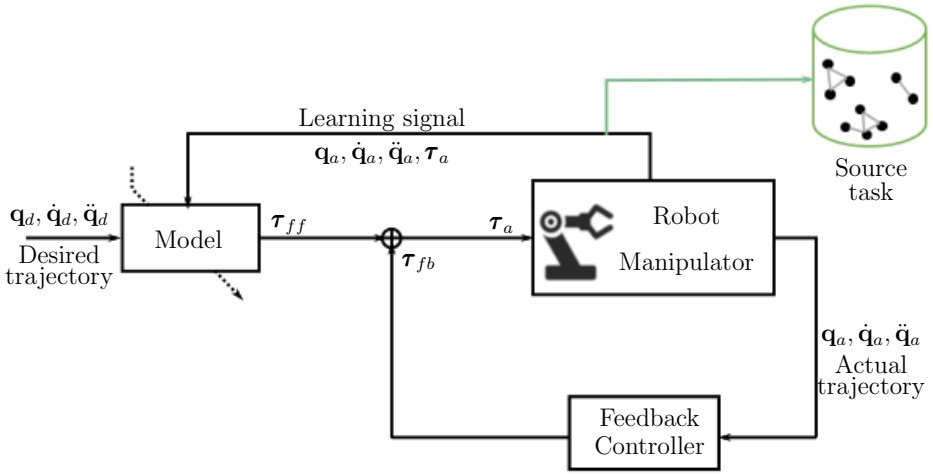


Figure 5-1: A learning-based control framework.

the early stages of learning, where the learned model is still poor, the predictions are inaccurate and the system relies heavily on the feedback term. This causes the actual states to differ from the desired states, and eventually, after many trials over the task, the model will improve and generalize to the desired states of the trajectory. The knowledge gained here is only used by the same robot when encountering the same task in the future.

When a new robot is available to learn the same task, it must go through the same laborious process, so as to collect training data. Our hypothesis is that, if its model is initialized offline, with data generated by a pre-existing robot while performing the same task (knowledge gained by the old robot), it may learn to perform the task in fewer trials. This may reduce training time of the new robot considerably, and is beneficial particularly in cases where operating the new robot is more expensive than operating the old robot. Next we outline how we propose applying knowledge transfer to improve learning of inverse dynamics.

5.4 Knowledge Transfer for Inverse Dynamics

We employ inter-robot transfer to improve learning a new task for target robot learner $\boldsymbol{\Omega}_t$ by re-using data generated from the experience of source robot learner $\boldsymbol{\Omega}_s$ when previously learning the same task. To achieve this, we learn a mapping function

$f : \boldsymbol{\chi}_s \mapsto \boldsymbol{\chi}_t$, for mapping samples from the domain $\boldsymbol{\chi}_s$ of the source robot to the domain $\boldsymbol{\chi}_t$ of the target robot. We assume training data $X_s \subset \boldsymbol{\chi}_s$ and $X_t \subset \boldsymbol{\chi}_t$ with correspondences, from which to learn f. This data set of correspondences is assumed to be generated by the source and target robots respectively (see Section 5.4.1). In our case of transfer for inverse dynamics, each data sample $\boldsymbol{x}_j^i = \{\boldsymbol{q}_a, \dot{\boldsymbol{q}}_a, \ddot{\boldsymbol{q}}_a, \boldsymbol{\tau}_a\}$, where j is either s or t for source data and target data respectively, $i = 1 : n$, and n is the number of samples in X_s and X_t.

We also assume that $\boldsymbol{\Omega}_s$ has previously learned the task at hand and thus has generated source task data $\boldsymbol{\xi}_s \in \Re^{m \times (4d_s)}$, where d_s is $\boldsymbol{\Omega}_s$'s DoF and m is the number of samples in the experience data[2]. Similar to X_s, $\boldsymbol{\xi}_s$ also contains joint positions, velocities, accelerations and torques associated with the task. Then we use the mapping f to transfer the source task $\boldsymbol{\xi}_s$ into the domain of $\boldsymbol{\Omega}_t$ to obtain the estimated target task data $\hat{\boldsymbol{\xi}}_t \in \Re^{m \times (4d_t)}$, where d_t is $\boldsymbol{\Omega}_t$'s DoF. Finally, $\hat{\boldsymbol{\xi}}_t$ is used to initialize $\boldsymbol{\Omega}_t$'s learning model offline, and the model is subsequently updated online as $\boldsymbol{\Omega}_t$ learns to perform the task.

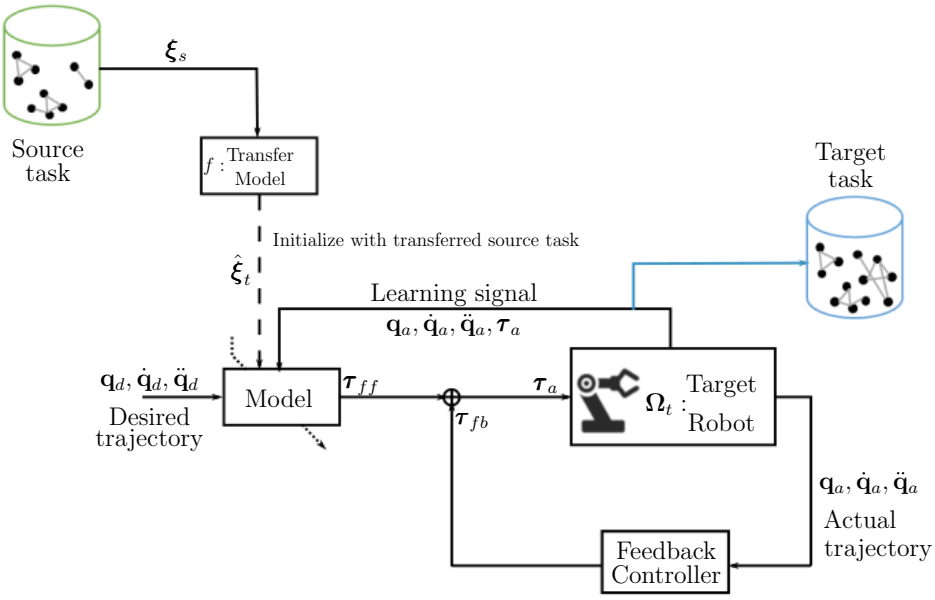


Figure 5-2: Our proposed transfer learning-based control framework.

Figure 5-2 illustrates our transfer learning-based control framework described

[2] m is actually the number of samples collected over many trials until $\boldsymbol{\Omega}_s$ learned to successfully perform the task.

above. We discuss the process of generating a sample of correspondences from the robots in Section 5.4.1 and algorithms for learning f from this sample in Section 5.4.2.

5.4.1 Collecting Correspondences

To train the transfer model f, we must collect correspondence data, X_s and X_t, from the robots. This means the robots must be controlled to generate similar movements, in order to identify correspondences. In [8] the authors used analytical controllers of the robots to track the same tasks with the same speed, and used cubic spline interpolation in the joint and torque spaces to obtain correspondences. We follow the same procedure; however, when learning a dynamic model for a real robot, we typically do not have access to its analytical controller, due to difficulties in accurately modeling its dynamic model. Thus, we employ PID controllers to track the same tasks with both robots.

We define correspondences in the task space of the robots, because the space is robot-agnostic, and also allows us to easily specify tasks. To obtain the correspondence data in joint and torque spaces, we assume kinematic models of the robots are available (either analytical or learned); we use them to map the tasks into the joint space, and use a PID controller to track them. We generate random straight-line trajectories of random length in the vicinity (in task space) of the tasks to be learned and track them with both robots with the same speed.

We do not necessarily require (nor expect) accurate tracking of the random tasks. The PID tracking control helps to generate correspondence data (approximately) locally from both robots. Our experiments demonstrate that correspondence data generated in this way is sufficient to learn transfer models. The random straight-line tasks could be replaced by any simple tasks (e.g. pick-and-place tasks) that both robots may be capable of tracking without complex dynamic models, and the data generated by both robots is then used to learn transfer models with which to accelerate learning of more complex tasks for the target robot.

Fig. 5-3 shows our framework for collecting samples of correspondences, where the straight-line random tasks have already been mapped into the joint spaces of the

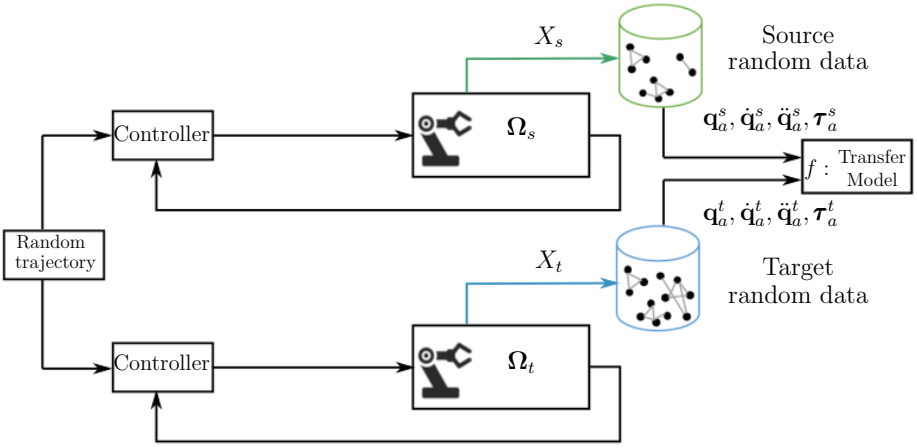


Figure 5-3: Our framework for collecting correspondence data.

robots using corresponding inverse kinematics (IK) models.

5.4.2 Learning the Transfer Model

In this section we describe learning of the transfer model f from a sample of correspondences generated from the robots as described in Section 5.4.1. We are provided with source domain data $X_s = \{\boldsymbol{x}_s^i\}_{i=1}^n$ and target domain data $X_t = \{\boldsymbol{x}_t^i\}_{i=1}^n$, where $\boldsymbol{x}_s \in \Re^{4d_s}$ and $\boldsymbol{x}_t \in \Re^{4d_t}$, and in general $d_s \neq d_t$ due to the robots potentially having differing DoFs. The aim is to learn the mapping f that we can use to transfer source domain data into the target domain, in such a way that is useful to the target robot. As previously mentioned in Section 5.2, this problem can be solved using manifold alignment techniques. Manifold alignment techniques allow for knowledge transfer between two seemingly disparate data sets, by aligning their underlying manifolds [149].

Any of the manifold alignment approaches discussed Section 3.1, Unsupervised Manifold Alignment [149], shared Autoencoders [53] and shared Gaussian Process Latent Variable Models [35], could be used in our framework to learn the transfer models. In our experiments the simple combination of PCA and PA proved sufficient, and we compared it against the combination of PCA and our LPA. For LPA, we cluster the data in the state space of the original source data (i.e. $\{\boldsymbol{q}_a, \dot{\boldsymbol{q}}_a\} \in \Re^{2d_s}$). By applying PCA, we assume a linear relationship between the original data spaces and

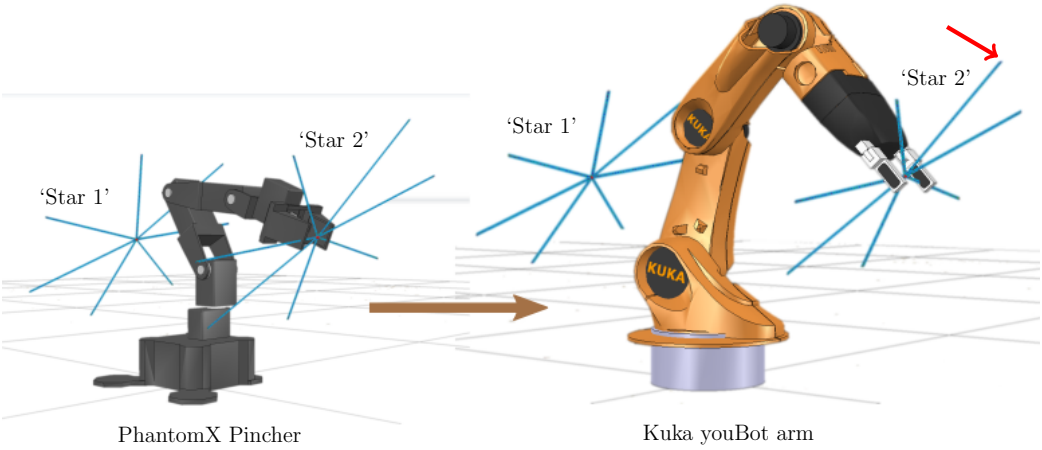


Figure 5-4: Example of knowledge transfer from a low-cost robot to a more expensive and heavier robot. The red arrow indicates a point that is challenging for the robot to reach at high speeds.

the corresponding latent representations. This is reasonable in our case because the training data is collected in the vicinity of the tasks to be transferred, and therefore the latent representations lie in relatively simple manifolds.

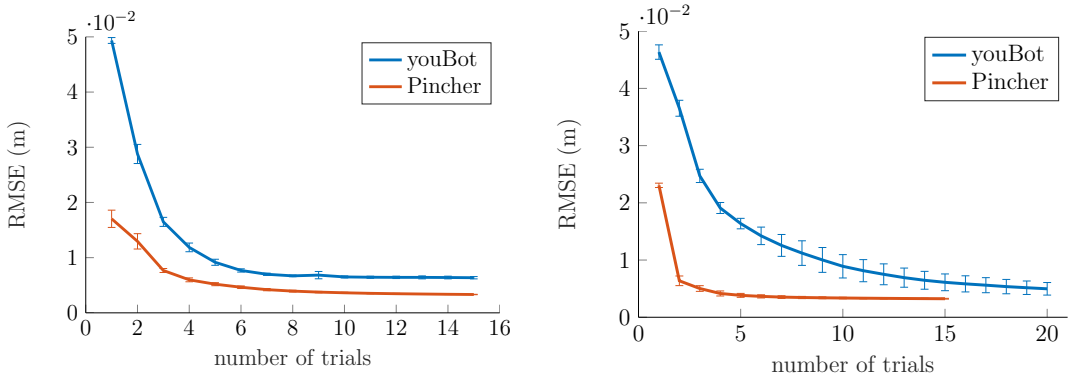
5.5 Experiments

5.5.1 Experimental Setup

We conducted experiments in simulation to transfer knowledge between a 5-DoF arm of the Kuka youBot and the 4-DoF Interbotix PhantomX Pincher arm (see Fig. 5-4). Both robots were simulated in V-REP[3] and controlled by a Matlab script via a V-REP remote application interface. The two robots have different kinematic and dynamic properties[4]. The Pincher arm is smaller, lighter and has a low torque rating on its joints – limited to 2.5 Nm, whereas the youBot arm is bigger, heavier and has a relatively higher torque rating on its joints – limited to 100 Nm.

As benchmark tasks, the robots learn to track the position of two 'star-like' figures placed at different locations in the task spaces of the robots and in different

[3]http://www.coppeliarobotics.com/
[4]We used default properties in V-REP 3.3.2.



(a) 'Star 1' task.

(b) 'Star 2' task.

Figure 5-5: Learning progress of the tasks by the robots.

orientations, as shown in Fig. 5-4. This 'star-like' trajectory has components of high acceleration, which makes tracking difficult, and is often considered in human motor control experiments and in robot control. Each 'star' task is composed of 7 straight lines starting from the center and pointing outwards.

The robots are required to perform the tasks by following each straight line starting from the center, going outwards and coming back to the center before following the next straight line. Each straight line must be tracked for 1.2 seconds[5] in total (0.6 seconds from the center to the end and another 0.6 seconds back to the center). Therefore one trial of tracking each task takes 8.4 seconds. The two tasks are denoted 'Star 1' and 'Star 2' as shown in Fig. 5-4.

5.5.2 Learning Inverse Dynamics Model

The aim of this experiment is to analyze online learning of the inverse dynamics model for tracking control applications and highlight the difference in learning performance due to the different dynamic properties of the robots. We employ LWPR [146] for learning the inverse dynamics model. We tuned hyper-parameters *init_D* and *init_alpha* using a grid strategy, following the guideline from the LWPR software package[6], and found $init\_D = 30$ for both robots to be sufficient, and $init\_alpha = 1.1$ for the Pincher arm and $init\_alpha = 0.01$ for the youBot arm.

[5]This is simulated time in V-REP.

[6]http://wcms.inf.ed.ac.uk/ipab/slmc/research/software-lwpr

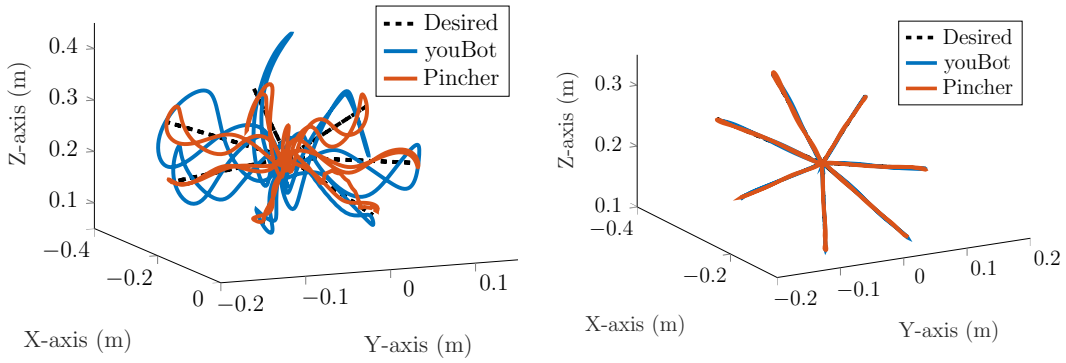
The following PID gains were used in all experiments: $K_p = 5\boldsymbol{I}$, $K_d = 0.1\boldsymbol{I}$ and $K_i = 0.01\boldsymbol{I}$ for the Pincher arm, and $K_p = 100\boldsymbol{I}$, $K_d = 2.0\boldsymbol{I}$ and $K_i = 0.01\boldsymbol{I}$ for the youBot arm, where $\boldsymbol{I}$ is an identity matrix whose size corresponds to robot DoFs. The learning procedure is as outlined in Section 5.3. Predictions and model updates are done at 100Hz for both robots, which is the current frequency at which to obtain stable torque control via a remote client in V-REP.

Figure 5-5 shows the learning performance of the tasks by the robots, where Fig. 5-5a shows results for 'Star 1' and Fig. 5-5b for 'Star 2'. The performance is measured in terms of the average error in each trial, of tracking the position of the desired tasks at each time step by the end-effector of the robot in the task space. We conducted 10 runs for each experiment and reported the mean and standard deviations of the tracking error.

We observe that the robots successfully learn to track the tasks over time, measured in terms of number of trials, indicated by the decreasing tracking errors. The smaller and lighter Pincher robot learns significantly faster and better, as it converges to lower tracking errors in fewer trials. The youBot robot converges slowly, especially for 'Star 2' where it requires more than 20 trials. The youBot arm requires larger PID gains due to its heavier components requiring larger torques to move, resulting in large feedback torques, especially in the first few trials where the learned model is still poor. For instance, in the first learning trial, the second joints of the robots exert the largest torque values, with the youBot arm exerting torque between -26 and 39 Nm and the Pincher robot exerting between -0.5 and 1.6 Nm.

Figure 5-6 shows example end-effector trajectories of both robots when tracking 'Star 2', overlaid on top of the desired trajectory (black dashed line). In the first learning trial (see Fig. 5-6a), tracking for both robots is poor, with the youBot robot performing the least. This may turn the robots into a danger to their environments and themselves, especially when interacting with objects.

We noticed that the youBot accumulates large errors when attempting to execute motions that require it to almost fully stretch out and raise its end-effector high at high speeds (see red arrow in Fig. 5-4), probably because these motions drive the robot



(a) First trials.

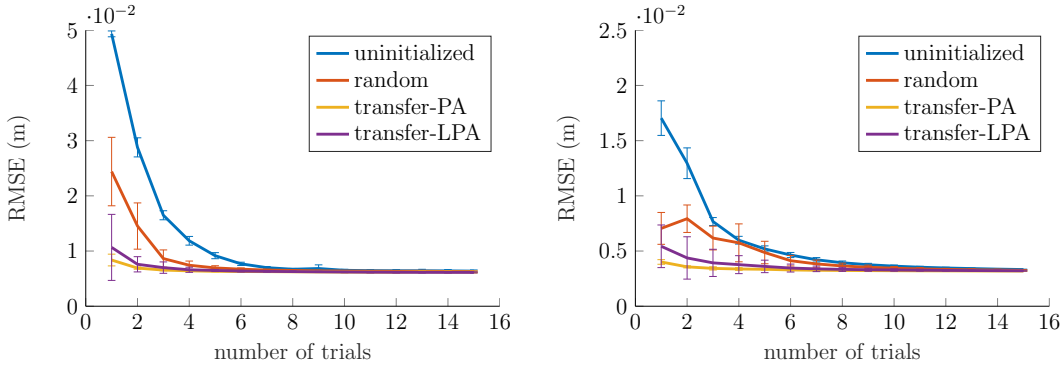
(b) Last trials.

Figure 5-6: 'Star 2' progress in the first and last trials.

far from its center of mass. This makes 'Star 2' harder to learn. The same applies to the Pincher robot, but to a lesser extent, probably due to its lighter materials. Nevertheless, both robots eventually learn to accurately track the tasks, as shown in Fig. 5-6b, where, in the last learning trial, the robots no longer unnecessarily exert too large torque values. Compared to the first learning trial, the second joints of the robots exert smaller torque values in the last few trials, with the youBot arm exerting torque between -0.9 and 15 Nm, and the Pincher arm exerting torque between -0.12 and 0.8 Nm.

5.5.3 Transfer for Inverse Dynamics Model

The aim of this experiment is to evaluate our knowledge transfer approach and to investigate the benefit of knowledge transfer in accelerating online learning of the inverse dynamics model. The learning procedure of the inverse dynamics model and parameters are the same as the previous experiment. The procedure for generating the training data for learning transfer models is as outlined in Section 5.4.1. We used random straight-line trajectories of random length and of duration 0.6 seconds each for motor babbling, requiring to be tracked at different speeds. The robots are controlled with PID controllers (parameters the same as before) to track these trajectories roughly in the vicinity of 'Star 1' and 'Star 2', for about 7 seconds per task. This means the motor babbling session lasts for about 14 seconds in total per



(a) Transfer to youBot.

(b) Transfer to Pincher.

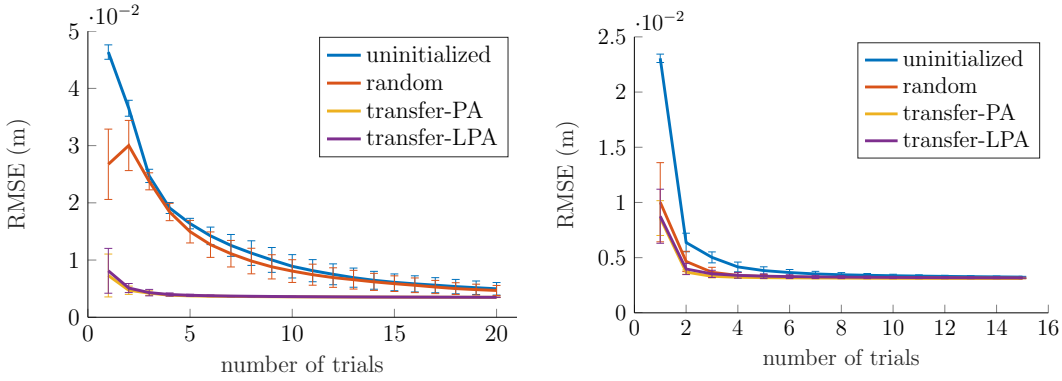
Figure 5-7: Accelerating 'Star 1' learning.

robot, resulting in samples of correspondences, X_{youbot} and $X_{pincher}$, with 1464 data points each.

The dimension of the youBot data is 20 ($\{\boldsymbol{q}_a, \dot{\boldsymbol{q}}_a, \ddot{\boldsymbol{q}}_a, \boldsymbol{\tau}_a\}$ for 5 DoFs), and that of the Pincher arm is 16 (same variables for 4 DoFs). We learned transfer models from this data, using PA and LPA, both combined with PCA for matching the dimensions of the data sets (see Section 5.4.2). We found the latent dimension $d = 16$ to be sufficient for both PA and LPA. Lower values of d lead to decreased transfer performance and values less than 10 barely transferred any useful knowledge.

We transferred to both robots, where the robots exchange roles of being source and target. We took the data generated by the source robot when learning a task from scratch and transferred it to the target robot domain. This provided us with additional data (12600 points for 15 trials and 16800 for 20 trials) that we use to initialize the target robot model. We denote initializing with transfer 'transfer-PA' and 'transfer-LPA' for PA and LPA respectively. In addition to evaluating the two transfer models, we also separately initialize the target robot model with target random data generated in the motor babbling session, which is denoted 'random', as it has previously been shown to accelerate learning [33, 105].

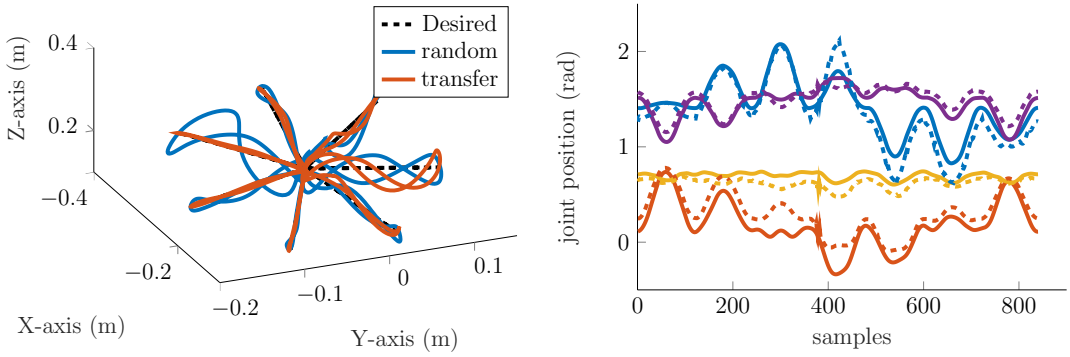
Figure 5-7 and 5-8 show results for initializing the target model offline, for 'Star 1' and 'Star 2' respectively, compared to learning from scratch (denoted 'uninitialized'). Knowledge transfer is accelerating learning considerably in most cases. In particular, when the youBot is the target learning 'Star 2' (see Fig. 5-8a), where learning



(a) Transfer to youBot.

(b) Tranfer to Pincher.

Figure 5-8: Accelerating 'Star 2' learning.



(a) First trial for 'Star 2'.

(b) First 4 joints of youBot.

Figure 5-9: Example transfer results for youBot.

from scratch and random initialization failed to converge within 20 trials, transfer converged within 5 trials. Both transfer models perform well, with PA slightly better than LPA in the early trials of 'Star 1'. This indicates that linear mappings are sufficient to transfer useful knowledge in this case.

In the general case of robots learning multiple tasks, transfer saves more time because the robots need only to generate motor babbling data once in the beginning. In the case of transferring to the youBot arm, learning from scratch for 'Star 1' converges within 9 trials (taking 75.6 seconds) and with knowledge transfer it converges within 4 trials (taking 33.6 seconds). Combining with the figures of 'Star 2', 20 trials (168 seconds) for learning from scratch and 5 trials (42 seconds) with transfer, and considering the 14 seconds for motor babbling, knowledge transfer saves up to 63% of training time in total, whereas random initialization only saves up to 4%. Fur-

thermore, the benefit of knowledge transfer would increase when more tasks must be learned in the same task space, since transfer models need only to be learned once.

Figure 5-9a shows example end-effector trajectories for random initialization and transfer in the first learning trial of the youBot learning 'Star 2'. We observe that for the most part transfer leads to stable and safe learning already in the first trial. This is due to transfer biasing exploration into relevant spaces near the desired trajectory, thus resulting in more efficient exploration, rather than sparse random trajectories that are not guaranteed to follow the same distribution as the desired trajectories. In Fig. 5-9b, we show an example of mapping the joints of the Pincher into the youBot domain using PA, where the solid lines are youBot joints and the dashed lines are transferred Pincher joints. We can see that although the linearly transferred Pincher joints are not exactly aligned with the target joints, they are distributed similarly, making it possible for learning to be accelerated.

5.6 Discussion

The fact that Procrustes Analysis is performing as well as and at times better than its non-linear extension, Local Procrustes Analysis, indicates that simple linear mappings are sufficient in transferring knowledge for inverse dynamics for our robots. This is most likely because learning in this case is aided when exploration is biased towards relevant spaces, and does not require transfer to be perfect. Also, LPA has been shown to perform better than PA when supplied with more data, as it creates more local clusters adapting to the local structure of the data [81, 56]. However, since in this case near-perfect mappings are not required and only guiding exploration is sufficient to accelerate learning, we can successfully transfer useful knowledge with a linear mapping learned on very few data points.

Knowledge transfer proved particularly beneficial for the youBot robot for which it is relatively more difficult to learn the tasks from scratch. The two robots we used are a good example of knowledge transfer between robots where the source robot is easier and cheaper to control than the target robot. The Pincher arm costs 40 times

less than the youBot arm, and has shown in our experiments to be able to learn the two tasks faster. Transfer enables experimenting with the cheaper robot and transfer to the expensive one.

Our simple PID control based data collection scheme proved to work well for generating data for learning transfer models in simulation. However, this needs to be validated on real robots and may prove difficult for bigger and heavier robots as they would require tuning of large PID gains. We employed LWPR as our model learning algorithm, mainly because its implementation was readily available online. It is possible that our results are biased by our choice of model learning algorithm. So an unbiased work must look at employing several other model learning algorithms discussed in Section 5.2, more especially recent ones, to ensure the conclusions drawn are not biased by a specific model learning technique.

Lastly, more experiments still need to be conducted on multiple robots to investigate cases in which transfer fails. The work in [111, 110, 54] is a step towards this goal, where the aim is to theoretically analyze transfer, however the theory has so far only been validated for single-input single-output control systems.

Chapter 6

Learning Sensorimotor Mappings

In robot modeling and control, kinematics modeling plays an important role. A robot kinematics model describes the motion of the robot with respect to some fixed reference frame, usually at the base of the robot, without considering any of the dynamic effects (e.g., forces and torques acting on the robot body, acceleration, etc). It allows, for example, a robot to be able make quick predictions about whether it can reach for objects in its surruounding environment using its hands. For motions decribed in the task space of the robot, a robot kinematics model is required to map them into the joint space, where control typically takes place (e.g., see Chapter 5).

Kinematics modeling also describes the relative positions and orientations (poses) between all the robot parts as well as with respect to the global reference frame. For example, the knowledge of poses of sensors located on the robot's hands, and the knowledge of the location of the hands on the robot's kinematics model, enables the transformation of the sensor data into a single reference frame for processing while the robot is in motion.

Conventionally, robot kinematics models are designed analytically by an engineer, using parameters of the robot released by the manufacturer, such as link dimensions, number of degrees-of-freedom (DoFs), the configuration of the joints and the connections of the links. If these parameters are not released by the manufacturer the engineer must estimate them through analysis of the robot's structure, which can cause modeling errors, requiring further calibration [80]. Analytical modeling tech-

niques fail when the exact values of such parameters are not available, such as in the case of complex robots with non-rigid links, flexible joints and when uncalibrated sensors provide noisy measurements. They are also not suitable for robots whose bodies change over time, potentially as a result of modification, repair, or material damage [130], as the static analytical model would need to be updated every time the characteristics of the robot change.

In such cases, machine learning techniques are employed as an alternative, where robot kinematics models are learned from data generated by the robot through its sensors and actuators. However, to successfully learn kinematics models a large amount of data must be collected from the robots, and this can be a time-intensive process for manipulators and humanoids with high DoFs and large state spaces. This is considerably worse in a multi-robot case, where each robot must learn its own models from scratch. In this chapter we investigate the improvement of learning kinematics models in the multi-robot setting using knowledge transfer, where kinematic data generated by pre-existing robots is used to accelerate learning for new robots.

The rest of the chapter is organized as follows. Section 6.1 provides an introduction to robot kinematics learning, and briefly reviews model learning techniques, and exploration techniques from developmental learning. Section 6.2 provides a review of accelerating kinematics learning, including techniques based on knowledge transfer. Section 6.3 describes the general problem of learning kinematics from real robot data and in Section 6.4 we present our knowledge transfer algorithm for accelerating the learning process. Section 6.5 provides our preliminary results validating our algorithm in simulations and we end the chapter with a discussion of our results and recommendation for future work in Section 6.6.

6.1 Introduction

Learning robot models from data is typically employed as an alternative to manual programming when the physical parameters of the robot are unknown or inaccurate. Unknown non-linearities can be taken into account as the model is estimated directly

from measured data, while they are typically neglected by the standard analytical modeling techniques [95]. Furthermore, modern robot systems are complex to model manually and are becoming increasing more diverse and widespread. As a result, learning of robot kinematics models has attracted much interest and has been used successfully in recent years [95, 11].

Robot kinematic models that are typically learned are forward kinematics (FK), which predicts the pose of the end-effector in the sensory space – the task space as perceived by sensors tracking the end-effector – given the robot joint angles in the motor space, and inverse kinematics (IK), which predicts the motor command (joint angles) required for the end-effector to reach a desired pose (e.g., to reach for objects). Learning these models involves learning of complex sensorimotor mappings, often in high-dimensional spaces.

Kinematics learning has been approached from mainly two angles that compliment each other. In model learning the focus is on developing machine learning algorithms that can learn robot models from large amount of data in high-dimensional spaces. Online and incremental learning algorithms are highly desired as they allow the robot to adapt to changes in the environment and/or the robot itself, and as the data is generated on the fly while the robot is in operation. Advanced statistical models have been developed here (e.g., [40, 146, 11, 22]). However, typically an assumption is made that good quality data is available in abundance and the algorithms perform well depending on the quality of the data sets.

Developmental robotics on the other hand attempts to study robotics from the perspective of building capabilities progressively via embodied interaction with the physical world. Its focus has been on designing exploration mechanisms that equip robots with the capability to autonomously explore their surrounding environment in order to collect data from which to learn the sensorimotor mappings. This has led to several strategies for exploration, ranging from exploration in motor space, referred to as motor babbling [36] (e.g., exploring the joint angles of a robot), to exploration in sensory space, referred to as goal babbling [114, 115] (e.g., exploring in end-effector space of a robot). Robots can explore these spaces randomly – random

motor babbling and random goal babbling – or they can explore actively, making use of the so-called interest models (also known as acquisition functions in the active learning area) to explore in such a way that maximizes some measure of progress – active motor babbling [120] and active goal babbling [6].

In general, when robot data is available, learning forward kinematics is straightforward, as the mapping from the motor space of the robot to its sensory space is one-to-one, and can be learned using standard regression techniques. Learning inverse kinematics on the other hand is an ill-posed problem for redundant systems, where the dimension of the sensory space m is less than the number of motors d (i.e., manipulator joints or DoFs). In such cases there usually are multiple joint configurations that lead to the same end-effector pose. As a result, standard regression techniques are not applicable.

Several model learning algorithms for inverse kinematics have been proposed in the literature, including those based on neural networks models [43, 70, 71] and statistical models [40, 22]. To overcome the non-uniqueness of the inverse kinematics solution, several techniques have been proposed, including learning by local optimization [10, 9], learning on the velocity level [40, 104, 39], learning with modular neural networks [99] and learning with structured prediction [11].

Complementing advanced model learning techniques with exploration mechanisms equips individual robots with the capabilities for autonomous exploration and learning of their sensorimotor mappings. In the single-robot case, exploration in the world is performed alone and the robot explores its own capabilities. In a multi-robot case, it may be beneficial for the robots to be able to share the knowledge they have acquired through their individual exploration. Such knowledge sharing has the potential to speed up development significantly and can allow more experienced or capable robots to impart their wisdom to others.

We explore knowledge transfer in the context of learning kinematics models, where an *experienced* robot shares its kinematic data with a new robot that is autonomously exploring its environment, in order to accelerate its learning process. By sharing kinematic data across robots, we show that the sensorimotor models of the new robot

can converge faster and also achieve a higher performance compared to individual exploration from scratch, when allocated the same exploration time. Next we review related work in improving learning of kinematics models.

6.2 Related Work

Very few work attempt to transfer knowledge across robots for accelerating learning of kinematics models, more specifically inverse kinematics. In [8], Procrustes Analysis was employed to transfer knowledge for learning forward kinematics, where data was generated using analytical models of the robots. In [23], an approach for transferring skills from human demonstrations for learning inverse kinematics of a soft-tendon driven manipulator was proposed. Reinforcement learning techniques were used to improve the transferred skills. Their targeted application was minimally invasive surgical tasks.

In this work we aim to transfer knowledge across developmental learning robots. Work related to ours in developmental robotics in is social robotics, where robots interact and learn from humans [92] or socially interact with each other [24]. However, none of the work in social robotics has been applied to transfer knowledge for learning kinematics models, more specifically inverse kinematics. Thus in this chapter we present, to the best of our knowledge, the first attempt to transfer knowledge across developmentally learning robots for inverse kinematics.

6.3 Problem Statement

We consider the relation between the motor commands $\boldsymbol{q} \in Q \subset \Re^d$ and their consequences in the sensory space $\boldsymbol{x} \in X \subset \Re^m$ (e.g., the position of the hand), where d is the number of DoFs and m is the dimension of the sensory space (e.g., $n = 3$ for the 3D spatial position of the hand). The forward kinematics function $L(\boldsymbol{q}) = \boldsymbol{x}$ describes the unique mapping from the motor space to the sensory space. It can be used to predict the consequences $\boldsymbol{x}^\star$ of setting the joints to some position $\boldsymbol{q}^\star$. For

robot control, an inverse function $L^{-1}(\boldsymbol{x}) = \boldsymbol{q}$ is required to predict the control command $\boldsymbol{q}^\star$ required to place the robot end-effector at some desired position $\boldsymbol{x}^\star$. The two functions model the sensorimotor mappings of the agent, and we will refer to them as sensorimotor models.

In general, the inverse function L^{-1} is complex and not uniquely defined if the number of DoFs d exceeds the dimension of the sensory space m. This is due to the existence of infinitely many solutions to $L^{-1}(\boldsymbol{x})$ when $m < d$. Several learning schemes for learning L^{-1} have been proposed, such as those discussed in Section 6.1. The robotic agent must learn such sensorimotor models by collecting samples $(\boldsymbol{q}^{(i)}, \boldsymbol{x}^{(i)})$ through its interaction with the environment, i.e., by executing motor commands $\boldsymbol{q}$ and observing the sensory consequences $\boldsymbol{x}$. The samples are then used to update the sensorimotor models $\hat{L}$ and $\hat{L}^{-1}$.

One of the challenges in learning $\hat{L}$ and $\hat{L}^{-1}$ from autonomous robot exploration is that since Q and X can be high dimensional, exploration can be a long process. It has been shown that goal babbling – exploration in the sensory space X – can learn inverse models for redundant systems more efficiently, compared to motor babbling – exploration in the motor space Q, since the sensory space is usually of much lower dimensionality than the motor space for manipulators [114, 115, 6, 89]. Furthermore, efficiency in exploration has been shown to further improve when leveraging active learning techniques, where the robot explores in such a way to maximize some measure of learning progress [6, 89].

Here we explore accelerating exploration and learning with knowledge transfer, where an experienced robot shares its kinematics data with a new robot. The shared data is used to initialize the sensorimotor models of the new robot such that its learning is bootstrapped and accelerated. In this preliminary study we employ a simple random goal exploration mechanism and a simple model learning algorithm for learning the sensorimotor models, in order demonstrate the possibility and benefit of knowledge transfer. In the following sections we describe an online goal babbling scheme we employ for autonomous exploration (Section 6.3.1) and a simple algorithm for learning sensorimotor models (Section 6.3.2).

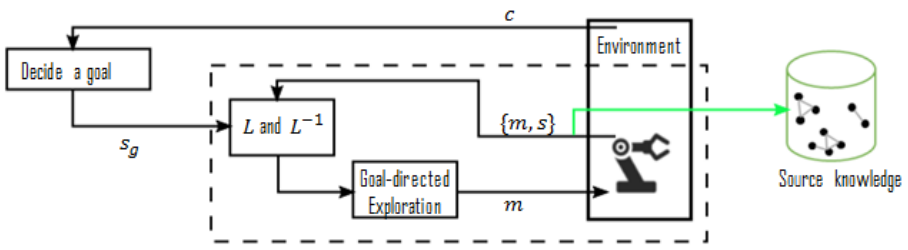


Figure 6-1: An illustration of a goal babbling framework. Based on an environmental context, a high-level learning architecture decides the next goal that a learning agent must attempt to reach. Without prior knowledge of the inverse model L^{-1}, the agent explores its environment by executing random motor commands and observing the consequences, and use the generated data to update its models L and L^{-1}. This process repeats until the agent is competent at reaching the goals and has gained knowledge about its internal models that it will use in the future when encountering the same goals.

6.3.1 Online Goal Babbling

In goal babbling, the robot randomly explores its sensory space X, by choosing random goals $\boldsymbol{x}_g$, and employs the current estimate $\hat{L}^{-1}$ of the inverse model to predict the command $\hat{\boldsymbol{q}}_g$ required to reach $\boldsymbol{x}_g$ with its end-effector. The robot then executes the predicted command and observes the sensory consequences, generating the training pair $(\boldsymbol{q}^{(i)}, \boldsymbol{x}^{(i)})$. Note that in the early stages of learning, the motor command prediction $\hat{\boldsymbol{q}}_g$ from the estimate $\hat{L}^{-1}$ will be poor such that when applied the robot may end up at $\boldsymbol{x}^{(i)} \neq \boldsymbol{x}_g$. A general exploration and learning framework is illustrated in Fig. 6-1. Typically, a learning algorithm is embedded in the system that incrementally learns $\hat{L}^{-1}$ as the robot explores and generates data, and also guides exploration by predicting the motor input for the next selected target to be explored. A forward model $\hat{L}$ can also be learned if needed from the generated data, but is not required by the goal babbling strategy, except for cases where the forward model is needed to estimate the inverse model, such as in Jacobian based inverse models or analytically inverting the forward model.

For high-dimensional redundant systems, the goal babbling strategy allows the robot to cover the sensory space more efficiently, avoiding wasting time in redundant parts of the motor space, as in motor babbling, which may execute many commands

that actually reach the same goal [114, 89]. Exploration is initiated near some home configuration $\boldsymbol{q}^{home}$, which results in $\hat{L}^{-1}$ learning to predict commands locally around $\boldsymbol{q}^{home}$, and over time exploration gradually moves away from $\boldsymbol{q}^{home}$ towards more complex movements, such as movements that contain multiple commands $\boldsymbol{q}$ that result in the same goal $\boldsymbol{x}_g$. This strategy biases robot exploration into focusing on learning simple movements that reach all the goals in the end-effector space and refining the movements at a later stage.

Algorithm 6 Goal Babbling

IN: End-effector space bounds $X_{bounds} = \{\boldsymbol{x}_{min}, \boldsymbol{x}_{max}\}$
IN: Time T for reaching selected goals
IN: Home configuration $\boldsymbol{q}^{home}$
Initialize learner: $L_{\boldsymbol{\theta}}^{-1} \leftarrow L_{\boldsymbol{\theta}_0}^{-1}$
for Number of epochs **do**
    Select a target $\boldsymbol{x}^{\star}$ randomly from X_{bounds}
    Select a sequence of targets towards $\boldsymbol{x}^{\star}$: $\boldsymbol{x}_t, t = 1, \cdots, T$
    for $t \in [1, T]$ **do**
        Estimate motor command: $\hat{\boldsymbol{q}}_t = \hat{L}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}_t)$
        Generate exploratory noise: $E_t(\boldsymbol{x}^{\star})$
        Execute perturbed motor command $\hat{\boldsymbol{q}}_t + E_t(\boldsymbol{x}^{\star})$ on robot
        Update model with generated data: $\hat{L}_{\boldsymbol{\theta}}^{-1} \leftarrow train(\boldsymbol{q}^{(t)}, \boldsymbol{x}^{(t)}))$
    end for
end for
OUT: Learned inverse model $\hat{L}_{\boldsymbol{\theta}}^{-1}$

Here, we employ a simple online goal babbling scheme proposed in [115] and is summarized in Algorithm 6. The model L^{-1} is parameterized by $\boldsymbol{\theta}$, which is updated online from data generated by the robot, thus is $L_{\boldsymbol{\theta}}^{-1}$. As inputs, the algorithm must be provided with sensory space bounds X_{bounds}, the number of waypoints T on the path when reaching for selected random goals $\boldsymbol{x}_g$ and a home configuration $\boldsymbol{q}^{home}$. The inverse estimate is initialized with the home configuration: $\hat{L}_{\boldsymbol{\theta}_0}^{-1} \leftarrow train(\boldsymbol{q}^{home}, \boldsymbol{x}^{home}))$, such that it always predicts the home configuration in the beginning.

To explore the sensory space, random goals $\boldsymbol{x}^{\star}$ are interatively chosen from X_{bounds}, and the robot attempts to reach each goal from the current configuration ($\boldsymbol{q}^{home}$ in the beginning) via $T-1$ intermediate targets $\boldsymbol{x}_t$. The targets are defined by a linear sequence between the current robot position $\boldsymbol{x}_0$ and the goal $\boldsymbol{x}_g$: $\boldsymbol{x}_t^{\star} = \frac{T-t}{T} \cdot \boldsymbol{x}_g + \frac{t}{T} \cdot$

$\boldsymbol{x}_{t-1}$, where $t = 1, \cdots, T$ denotes the substeps within one movement. To reach the intermediate targets, the robot uses the current estimate of the inverse model to infer a motor command $\hat{\boldsymbol{q}}_t$. An exploratory noise $E_t(\boldsymbol{x}^\star)$ is added to the inferred motor command and the noisy command is sent to the robot. The parameters of the inverse estimate (or if needed, also of the forward estimate) are updated from the generated examples $\boldsymbol{q}^{(t)}, \boldsymbol{x}^{(t)}$).

The exploratory noise is added in order to find kinematic solutions for all goals in the sensory space. Following [115], we model the exploration noise as a randomly chosen linear function

$$E_t(\boldsymbol{x}^\star) = A_t \cdot \boldsymbol{x}^\star + \boldsymbol{b}_t, A_t \in \Re^{d \times m}, \boldsymbol{b}_t \in \Re^d. \tag{6.1}$$

6.3.2 Sensorimotor Models

6.4 Guided Exploration with Knowledge Transfer

In the single-robot case, exploration in the world is performed alone and the robot explores its own capabilities. In a multi-robot case, it may be beneficial for the robots to be able to share the knowledge they have acquired through their individual exploration. Such knowledge sharing has the potential to speed up development significantly and can allow more experienced or capable robots to impart their wisdom to others. We investigate knowledge transfer for accelerating learning in a context of learning forward and inverse sensorimotor mappings L and L^{-1}.

Formally, we consider an experienced source agent Ω_s that has learned its own sensorimotor models L_s and L_s^{-1}, and a less experienced target agent Ω_t that is yet to learn its own sensorimotor models L_t and L_t^{-1}. Our aim is to utilize source agent data $\xi_s = \{\boldsymbol{q}_s^{(i)}, \boldsymbol{x}_s^{(i)}\}_{i=1}^T$, generated by the source agent while learning its sensorimotor models or synthesized from its learned models, to initialize the parameters of the target agent sensorimotor models, so as to accelerate learning of the target agent. In general, the source and target agents have different embodiments, resulting in differing motor and/or sensory spaces, i.e., $d_s \neq d_t$ and $m_s \neq m_t$, and different data

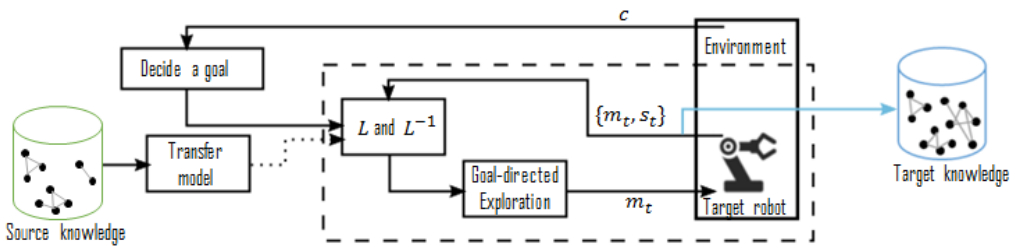


Figure 6-2: An illustration of our guided goal babbling framework. Based on an environmental context, a high-level learning architecture decides the next goal that a learning agent must attempt to reach. With the prior knowledge of the inverse model L^{-1} transferred from some source agent, the agent explores its environment by executing the transferred motor commands and observing the consequences, and use the generated data to update its models L and L^{-1}. This process repeats until the agent is competent at reaching the goals and has gained knowledge, which is obtained faster than learning from scratch.

distributions.

To effectively transfer the source data into the target agent domain, the source data must be configured such that it is useful to the target agent. Similar to other learning domains dealt with in this thesis, we learn a domain mapping f from samples of correspondences X_s and X_t generated by the robots. This domain mapping is then used to transfer source agent data ξ_s into the target agent domain to obtain estimated target agent data $\hat{\xi}_t$, which is subsequently used to initialize the parameters of the target agent sensorimotor models L_t and L_t^{-1}, as illustrated in Fig. 6-2.

To collect correspondences X_s and X_t, we propose an algorithm for guiding exploration of one robot with motor commands generated by another robot. This algorithm assumes we know correspondences between the motor spaces $\bar{\boldsymbol{q}}_s^{(i)}, \bar{\boldsymbol{q}}_t^{(i)} \in \bar{Q} \subset \Re^{d_c}$ of the robots, where $d_c \leq min(d_s, d_t)$ is the dimension of a motor subspace $\bar{Q}$ in which Ω_s and Ω_t have joints in common. In our proposed guided exploration algorithm, the target agent explores its environment for some period T_{guided}, using for example Alg. 6, executing only those motor commands $\bar{\boldsymbol{q}}_t^{(i)}$ that are in correspondence with the source agent, to collect X_t. Since $d_c \leq d_t$, the rest of the commands not in common with the source agent's stay at their 'home' values, i.e., in q^{home}.

The source agent Ω_s then executes the sequence of commands $\{\bar{\boldsymbol{q}}_s^{(i)}\}_{i=1}^{T_{guided}}$ to collect X_s corresponding to X_t, using Alg. 7. This guided exploration process is illustrated in

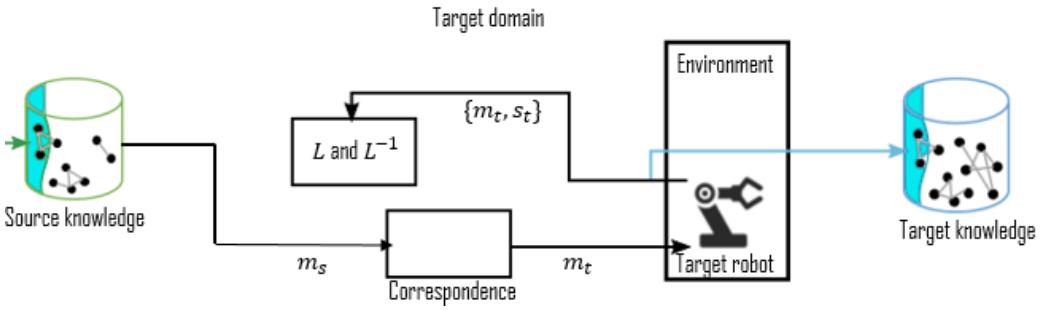


Figure 6-3: An illustration of our guided exploration framework. The source agent executes commands generated by the target agent in order to correspond its movements with those of the target agent and generate a sample of correspondences.

Fig. 6-3. The domain mapping f can then be learned from X_s and X_t as discussed in Section 6.4.1. The source agent experience, or synthesized, data ξ_s is transferred to the target agent domain to obtain estimated target agent data $\hat{\xi}_t$, which is subsequently used to initialize target agent sensorimotor models L_t and L_t^{-1}. The target agent then continues exploring autonomously using Alg. 6.

Algorithm 7 Guided exploration

IN: A sequence of corresponding Ω_t motor commands $\{\bar{\boldsymbol{q}}_s^{(i)}\}_{i=1}^{T_{guided}}$
$X_s = \{\}$
for $i \in [1, T_{guided}]$ **do**
    Execute motor command $\bar{\boldsymbol{q}}_s^{(i)}$ on Ω_s
    $X_s = X_s \cup \{\bar{\boldsymbol{q}}_s^{(i)}, \boldsymbol{x}_s^{(i)}\}$
end for
OUT: Source correspondence sample X_s

6.4.1 Transfer Models

The samples of correspondences $\{\bar{\boldsymbol{q}}_s^{(i)}, \bar{\boldsymbol{x}}_s^{(i)}\}_{i=1}^{T_{guided}}, \{\bar{\boldsymbol{q}}_t^{(i)}, \bar{\boldsymbol{x}}_t^{(i)}\}_{i=1}^{T_{guided}} \subset \Re^{d_c+m_c}$ have the same dimensionality $d_c + m_c$ of the subspace in which Ω_s and Ω_t have common variables. For the 3D positions the source and target sensory spaces have the same original dimensionality, i.e., $\bar{\boldsymbol{x}}_s^{(i)}, \bar{\boldsymbol{x}}_t^{(i)} \in \Re^{m_c}$, $m_c = m_s = m_t$, and $d_c \leq min(d_s, d_t)$. The domain mapping f then learns the (non-linear) transformation of sensory signals $\boldsymbol{x}_s^{(i)}$ and $\boldsymbol{x}_t^{(i)}$ given correspondences in the motor subspace $\bar{Q}$.

The source experience data $\xi_s = \{\bar{\boldsymbol{q}}_s^{(i)}, \boldsymbol{x}_s^{(i)}\}_{i=1}^{T}$ is then transferred using f to ob-

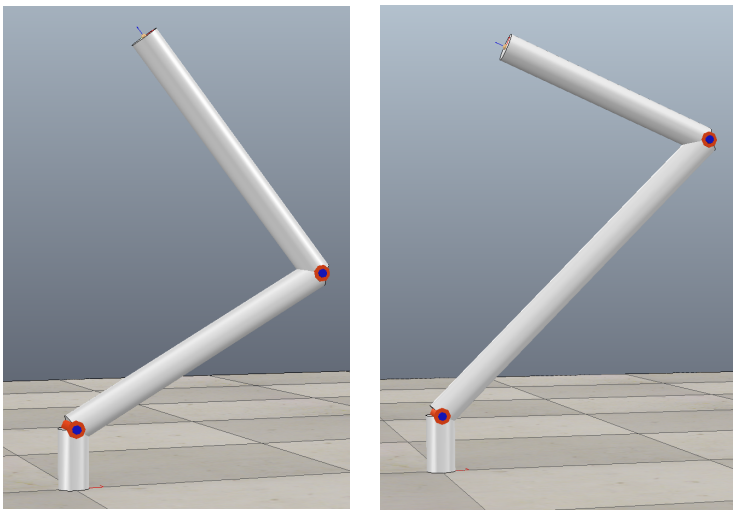
tain estimated target data $\hat{\xi}_t = \{\bar{\boldsymbol{q}}_t^{(i)}, \boldsymbol{x}_t^{(i)}\}_{i=1}^{T}$. Since the dimensionality d_c of the transferred motor data may not be the same as the dimensionality d_t of the target motor space, we pad the transferred data with those values in $q_t^{home} \notin \bar{Q}$. Then any manifold alignment technique can be used to learn f from $\{\bar{\boldsymbol{q}}_s^{(i)}, \bar{\boldsymbol{x}}_s^{(i)}\}_{i=1}^{T_{guided}}$ and $\{\bar{\boldsymbol{q}}_t^{(i)}, \bar{\boldsymbol{x}}_t^{(i)}\}_{i=1}^{T_{guided}}$. Employing Procrustes Analysis or Local Procrustes Analysis, points in ξ_s are transferred individually into the target domain using Alg. 2 or 4 respectively.

6.5 Experiments

In this section we present some preliminary results validating knowledge transfer for accelerating learning of inverse kinematics using two sets of experiments. In Section 6.5.1 we present results for transfer between two-link planar robots with 2D task spaces, to illustrate our approach and to compare transfer with Procrustes Analysis and Local Procrustes Analysis. In Section 6.5.2 we demonstrate the benefit of knowledge transfer in a realistic scenario, where we transfer knowledge between two humanoid robots. All robots are simulated in V-REP [113] and controlled via a remote Matlab script.

We assume the source robot has successfully learned its sensorimotor models and that we can synthesize its experience data. So to evaluate the benefit of knowledge transfer, we compare the time it takes for the target robot to learn from scratch and the time it takes to learn when provided with additional knowledge transferred from the source robot. We only conduct experiments for learning inverse kinematics, since learning forward kinematics is easier and exactly the same procedure for transfer is applicable. We employ a simple k-NN regression model as our sensorimotor model with $K = 3$.

For all learning setups, we evaluate the learning progress by testing the learned sensorimotor model at evenly spaced time intervals on some test data evenly distributed in the target robot's task space. We use two measures of progress. We calculate the average error of reaching all the test points and calculate the reaching rate as the ratio of the points reached within some error threshold – 0.01 m in Section



(a) Source robot. (b) Target robot.

Figure 6-4: Two-link robots used in the experiments.

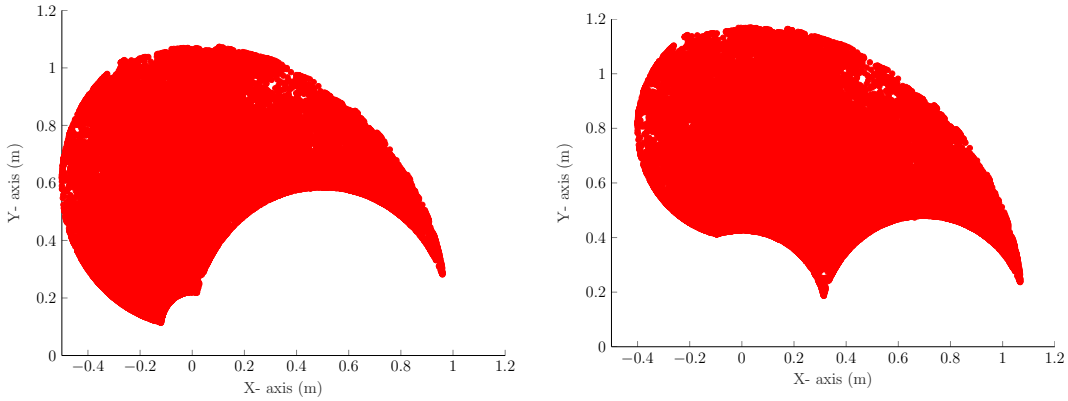
6.5.1 and 0.015 m in Section 6.5.2. All the experimental results are averaged over 10 runs.

6.5.1 Simple Two-link Planar Robots

In this experiment we analyze knowledge transfer between two two-link planar robots with differing link lengths and same joint limits. The goal of this experiment is to illustrate our knowledge transfer method for accelerating learning of inverse kinematics. The parameters of the two robots are shown in Table 6.1 and the two robots are shown in Fig. 6-4. The dimensionality of the motor and sensory spaces of both robots is 2, i.e., $\boldsymbol{q}_s, \boldsymbol{q}_t, \boldsymbol{x}_s, \boldsymbol{x}_t \in \Re^2$, and thus $X_s, X_t, \xi_s, \xi_t \in \Re^4$. The two robots share the same motor space, so correspondences are easily defined in their original motor spaces.

Parameter	Source	Target
`Link 1`	0.5	0.7
`Link 2`	0.5	0.4
`Motor 1`	$[-\pi/2, 0]$	$[-\pi/2, 0]$
`Motor 2`	$[0.4, 2.9]$	$[0.4, 2.9]$

Table 6.1: Parameters (lengths (m) and joint limits (rad)) of two-link robots.



(a) Source robot.

(b) Target robot.

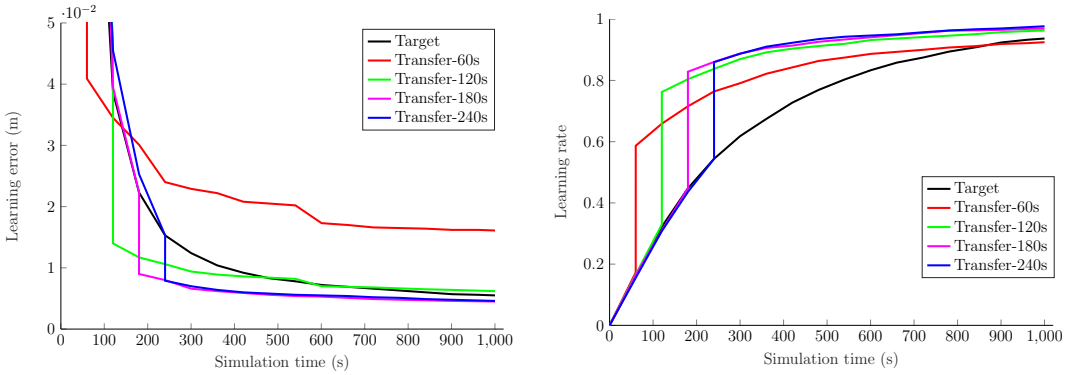
Figure 6-5: 2D task spaces of the robots.

The differences in the link lengths of the two robots make direct transfer of ξ_s into the target domain infeasible, as the distributions of their sensory spaces are different, as illustrated in Fig. 6-5. We employ Procrustes Analysis and Local Procrustes Analysis to transfer ξ_s into the target to obtain $\hat{\xi}_t$ to initialize the target robot sensorimotor model. For learning the target robot sensorimotor model, we perform goal babbling using Alg. 6 for 1000 seconds.

For learning with knowledge transfer, we perform guided exploration using Alg. 7 for 60, 120, 180 and 240 seconds to evaluate the effect of transfer in the early stages as well as in later stages of learning. After performing guided exploration and transfer from the source robot, the target robot continues to explore using Alg. 6 for the remainder of the time. For all learning setups, we evaluate the learning progress by testing the sensorimotor model on test points evenly distributed in the task space, at 60 seconds intervals.

Fig. 6-6 shows the average results of knowledge transfer with LPA, where Fig. 6-6a shows the reaching error and Fig. 6-6b shows the reaching rate. The black curve indicates the progress of the target robot learning from scratch, and the other curves indicate learning with knowledge transfer where the transfer model was learned and transfer applied at different intervals.

We observe that the learning progress is boosted instantly when the knowledge is transferred, and the target robot achieves higher learning rates compared to learning



(a) Reaching error.

(b) Reaching rate.

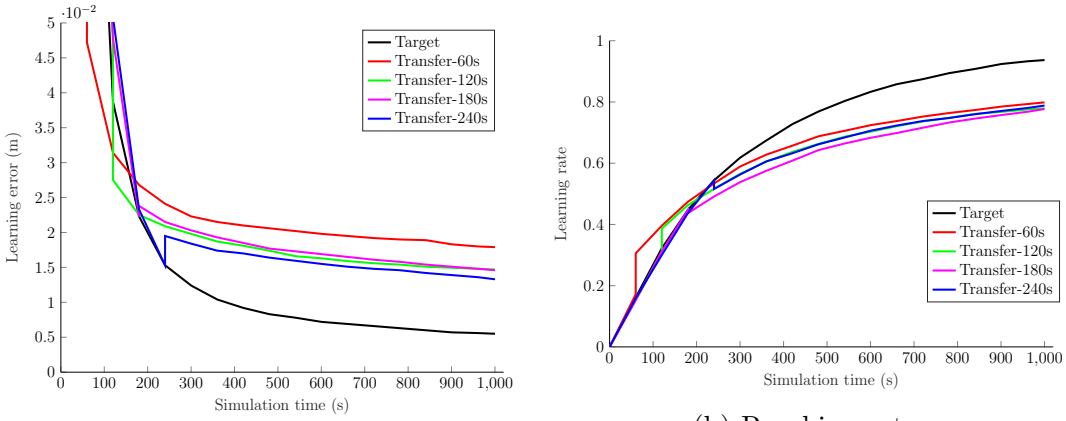
Figure 6-6: Knowledge transfer with Local Procrustes Analysis.

from scratch. However, transfer with LPA is beneficial with respect to the final learning performance when applied at 180 seconds and beyond. This is because at 180 seconds there is enough correspondence data collected by the target robot and LPA achieves a better transfer accuracy when given more data. When only a few data to learn an LPA transfer model is available, and LPA cannot learn a good model, negative transfer occurs.

Fig. 6-7 shows the average results of knowledge transfer with Procrustes Analysis. Knowledge transfer with Procrustes Analysis on the other hand fails to accelerate learning, with only a slight boost of progress when transfer is applied early. This is due to the linear mapping failing to capture the complex non-linearity of the robot data spaces. Providing Procrustes Analysis with more data does not improve its transfer accuracy, as has been demonstrated before in Section 3.6. Knowledge transfer with Procrustes Analysis in this case negatively affects the final performance as there are many data points that were inaccurately transferred.

6.5.2 Knowledge Transfer between Nao and Poppy

This experiment demonstrates the benefit of knowledge transfer in a realistic scenario, between two of the popular robots in developmental robotics. We transfer data from SoftBank's Nao robot to Poppy, shown in Fig. 6-8. Poppy is a 25-DoF humanoid robot with two 4-DoF arms. We use Poppy as our target robot and learn inverse



(a) Reaching error.

(b) Reaching rate.

Figure 6-7: Knowledge transfer with Procrustes Analysis.

kinematics for its right arm, where we allow the torso rotation about the Z axis, allowing it to turn left and right. This results in a 5-DoF motor space $\boldsymbol{q}_t \in \Re^5$ and a 3D sensory space $\boldsymbol{x}_t \in \Re^3$, and thus $X_t, \xi_t \in \Re^8$.

Our source robot, Nao, is a 25-DoF humanoid robot with two 5-DoF arms. The 25 DoFs include the motors for controlling its fingers. Nao's motor space is also 5-DoF $\boldsymbol{q}_s \in \Re^5$ and it also has a 3D sensory space $\boldsymbol{x}_s \in \Re^3$, and thus $X_s, \xi_s \in \Re^8$. Nao's arms have additional wrist joints compared to Poppy's 4-DoF arms but Nao does not have a motor that rotates its torso about the Z axis. So the robots have a $d_c = 4$ dimensional common motor space $\bar{\boldsymbol{q}}_s, \bar{\boldsymbol{q}}_t \in \bar{Q} \subset \Re^4$ and thus their correspondence data is in $\Re^7$. This means that since Nao cannot rotate its torso it cannot transfer knowledge about reaching using the torso movement, and since Poppy arms do not have wrist joints, Poppy cannot learn any knowledge about reaching using wrist movements.

So we aim to accelerate Poppy's learning by transferring Nao data generated by the 4 joints that it has in common with Poppy. Then Poppy continues exploring utilizing all the motors available, i.e., the 4 joints in common with Nao's right arm and the torso joint rotating about the Z axis. Poppy explores its environment using Alg. 6 for 5000 seconds to learn to reach within a grid defined by $X_{bounds} = [(0.1, 0.05, 0.4), (0.4, 0.3, 0.8)]$, and its learning progress is evaluated every 300 seconds on a grid of test points evenly distributed inside the 3D boundary as shown in Fig. 6-9.

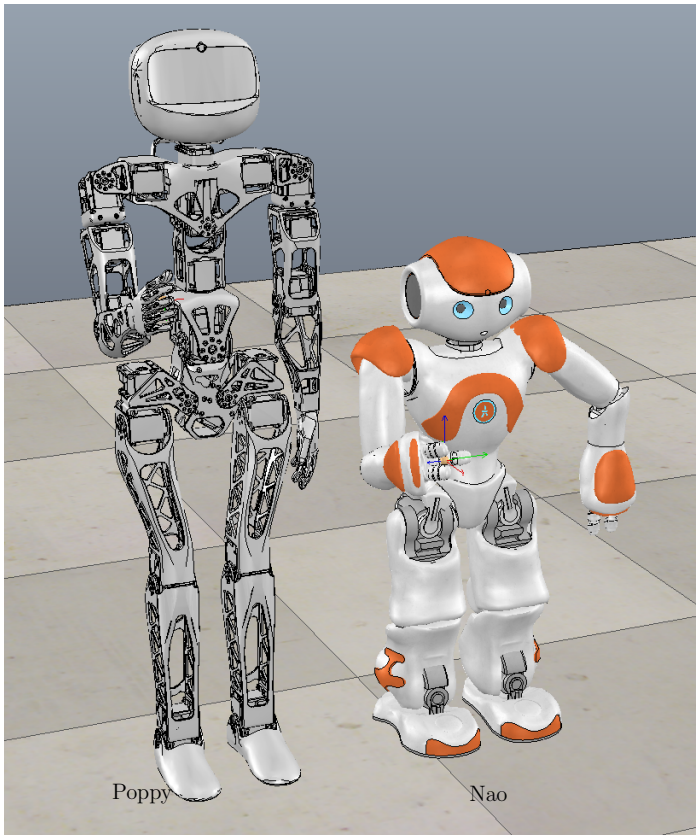


Figure 6-8: Example of knowledge transfer from a small Nao to a bigger Poppy robot. On the left is the Poppy robot and on the right Nao robot.

We perform guided exploration for 600 seconds and learn the transfer model using Local Procrustes Analysis to transfer Nao's experience data. Fig. 6-10 shows the results of knowledge transfer using LPA. The blue curve labeled 'Poppy' indicates the learning progress of Poppy from scratch, using all 5 motors. We observe that learning in this case varies a lot. This is more likely due to the redundancy introduced by the 5 motors, as there are many ways for Poppy to reach points inside the boundary. For example, it can reach some points either without moving its torso or with the torso.

The orange curve labeled 'Nao transfer' indicates the results of transferring Nao's experience data using LPA. We transferred Nao's experience equivalent to the amount of data Nao generates when learning its model from 600 seconds until 5000 seconds. This means when we transfer the knowledge to Poppy at 600 seconds, we are providing Poppy with additional 4400 seconds worth of knowledge and it continues exploring from 600 seconds until 5000 seconds. The additional data provided to Poppy instantly

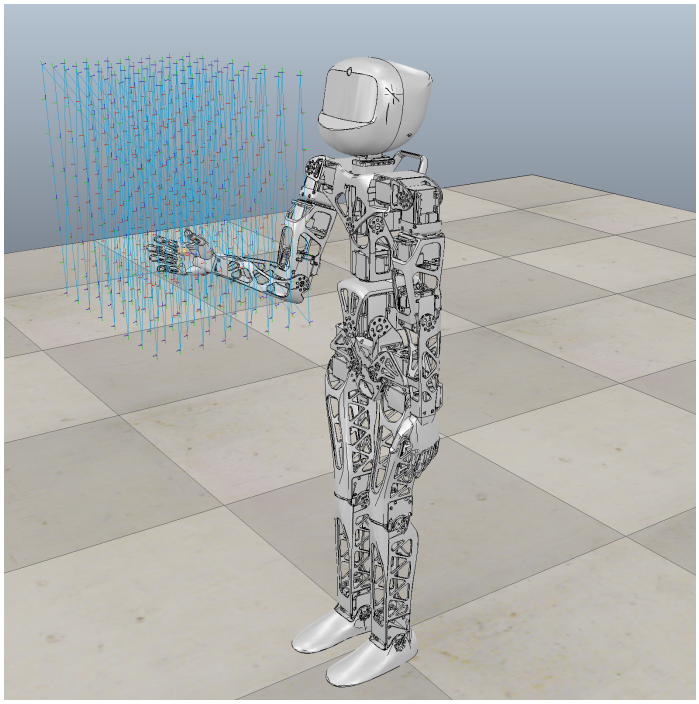
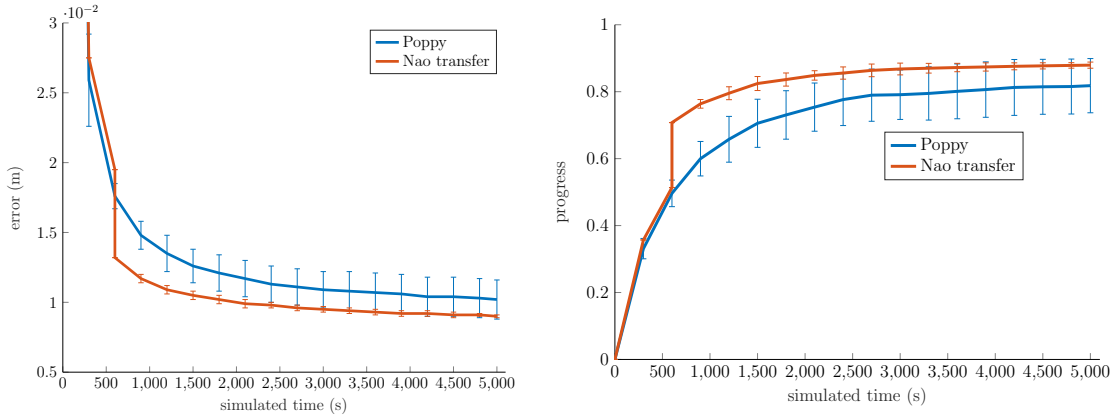


Figure 6-9: Poppy learning to reach inside a 3D boundary space.

boots its learning progress and helps it achieve faster learning and better final performance. Without the additional knowledge transferred by Nao, Poppy reaches a reaching rate of about 0.81 in 5000 seconds. With the help of the transferred knowledge, Poppy reaches a reaching rate of 0.81 in 1500 seconds and a final performance of 0.88. Thus transfer saves up to 3500 seconds if learning is stopped at 1500 seconds.

6.6 Discussion

Our preliminary results presented in the previous section demonstrate the possibility and benefit of knowledge transfer in accelerating learning of sensorimotor mappings. Results presented in Section 6.5.1 compared knowledge transfer with Procrustes Analysis and our proposed (non-linear) model, Local Procrustes Analysis. Results show that the linear mapping of Procrustes Analysis is limited to transfer kinematic data for accelerating a sensorimotor model learned using k-NN regression. Negative transfer occurs, which results in k-NN regression being computed on incorrect data, as indicated by the degradation in learning performance of the target robot after transfer.



(a) Reaching error.

(b) Reaching rate.

Figure 6-10: Knowledge transfer from Nao to Poppy with Local Procrustes Analysis.

Our proposed model proved particularly suitable to this application as it successfully accelerated learning for the toy example used. Negative transfer only occurs when LPA transfer is applied before enough data has been collected.

Results in Section 6.5.2 demonstrated the benefit of knowledge transfer using our proposed transfer model in a realistic scenario consisting of two humanoid robots. We demonstrated that the knowledge transferred from Nao to Poppy helps to save up to 70% of learning time compared to learning from scratch, and that knowledge transfer can also improve the final learning performance of the target robot.

We believe more experiments still need to be conducted with other robots to determine when transfer is possible and beneficial. Ideally we should be able to provide conditions in which transfer is guaranteed to be beneficial and conditions in which we know beforehand that transfer is not possible. We plan to conduct experiments to transfer knowledge from Poppy to Nao (in the opposite direction), and also introduce a third robot with different kinematics, such as the iCub, as future work. This should provide us with enough experimental data to analyze conditions where transfer is possible and beneficial.

The k-NN regression model employed for learning sensorimotor models may have a limitation when negative transfer occurs. This is because regression is performed on all data saved in memory and equal weights are assigned to all data. To reduce the effect of negative transfer, one may consider weighting transferred data less, espe-

cially when the robot is generating new data as it continues to explore. Alternatively, a parameterized model representations can be used, such as neural networks, polynomial regression, etc., whose parameters are updated when new data is generated. This has the potential that the parameters initialized with knowledge transfer may perform well in the early stages of learning and will get updated by new data and quickly adapt, thereby forgetting the inaccurately transferred knowledge.

Lastly, we employ a random goal babbling scheme in all our experiments. Given that active exploration schemes have been shown to learn sensorimotor models faster than random ones, since they seek to explore regions where the robot has yet to learn to successfully reach, it is reasonable to suggest that active exploration schemes would benefit from knowledge transfer more than random exploration schemes. We believe that an active learning agent would quickly shift its attention to areas where knowledge transfer was not effective, such as if negative transfer occurred or areas where the source robot could not reach, rather than to continue exploring randomly without considering the information from the transferred knowledge.

Chapter 7

Conclusion

In this thesis, we confronted some of the challenges encountered when robots share knowledge that they have learned through their individual experiences. The challenges we tackled are due to robots having different embodiments and physical characteristics. More specifically, the differences in terms of kinematics – robot dimensions, number of degrees-of-freedom, joint configurations and link connections – and dynamics – mass, center of mass, inertia matrix, etc. They result in the robots having different state and actions spaces – resulting in their machine learning models having different feature spaces – and different data distributions.

We are particularly interested in accelerating learning of low-level robot motor skills with knowledge transfer, where we re-use data gathered by pre-existing robots. We reviewed work in knowledge transfer in robot learning in Chapter 2 and related work in accelerating learning with knowledge transfer. Our work is amongst the few work applying transfer learning for accelerating kinematics and dynamics learning in the literature. We contribute a transfer learning model, Local Procrustes Analysis presented in Chapter 3, to enable the transfer of knowledge across robots with different embodiments and physical characteristics.

Our proposed transfer learning model is particularly suited for robotics applications where the data distributions require a non-linear mapping to be aligned, and we compared it against a popular linear model, which failed to accurately align the data. We applied our non-linear transfer learning model to transfer knowledge in

three different learning domains.

In Chapter 4 we proposed a knowledge transfer approach to robot learning from demonstrations which applies our transfer learning model. We demonstrated that our approach can reasonably initialize the parameters of a parameterized motor primitive and that the skills transferred from a human teacher can be reproduced by the robot. Furthermore, we showed that our method can be extended to allow one robot that has acquired knowledge from a human teacher to act as a teacher to another robot and transfer its knowledge. However, we only demonstrated the efficacy of transferring skills across robots and not the acceleration of learning motor skills. In future work we plan to conduct experiments to test our method in a realistic scenario where the robot learner improves the transferred skills using reinforcement learning techniques and evaluate whether our transfer learning approach can accelerate the learning process.

In Chapter 5 we proposed a knowledge transfer framework for accelerating online learning of inverse dynamics for manipulator control. We demonstrated that our approach can transfer knowledge between robots with different kinematic and dynamic properties, including different number of degrees-of-freedom. In contrast to previous work, we proposed an approach for collecting correspondences from the robots without using their analytical models – which is what is required in a realistic scenario. Our results showed that for inverse dynamics, the linear mapping of Procrustes Analysis is sufficient to transfer knowledge for accelerating learning of the target robot.

In Chapter 6 we proposed a guided exploration approach for knowledge transfer between robots for accelerating learning of sensorimotor mappings for a developmental learning robot. We evaluated our approach in accelerating learning of inverse kinematics between humanoid robots with different kinematics. Our preliminary results show that knowledge transfer using our proposed transfer learning model is beneficial. We believe our approach has the potential to transfer knowledge across many different humanoids and we plan to conduct more experiments in the future to ascertain that. In particular, we plan to use parameterized sensorimotor models that can quickly forget inaccurately transferred knowledge and employing active exploration strategies that will take more advantage of the transferred knowledge to guide

exploration towards unexplored regions. Lastly, we plan to investigate an approach that will allow robots to autonomously detect their correspondences.

Bibliography

[1] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew E. Taylor. Online Multi-Task Learning for Policy Gradient Methods. In *Proceedings of the 31st International Conferences on Machine Learning (ICML)*, June 2014. 25

[2] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew E. Taylor. Unsupervised Cross-Domain Transfer in Policy Gradient Reinforcement Learning via Manifold Alignment. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, January 2015. 27

[3] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, May 2009.

[4] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11(1):75–113, Feb 1997.

[5] P. Azad, T. Asfour, and R. Dillmann. Toward an unified representation for imitation of human motion on humanoids. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2558–2563, April 2007.

[6] Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

[7] John Blitzer, Dean Foster, and Sham Kakade. Domain adaptation with coupled subspaces. In *Conference on Artificial Intelligence and Statistics*, Fort Lauterdale, 2011.

[8] Botond Bócsi, Lehel Csató, and Jan Peters. Alignment-based transfer learning for robot models. In *Proceedings of the International Joint Conference on Neural Networks*, 2013.

[9] Botond Bócsi, Lehel Csató, and Jan Peters. Indirect robot model learning for tracking control. *Advanced Robotics*, 28(9):589–599, 2014.

[10] Botond Bócsi, Phillip Hennig, Lehel Csató, and Jan Peters. Learning tracking control with forward models. In *2012 IEEE International Conference on Robotics and Automation*, pages 259–264, May 2012.

[11] Botond Bócsi, Duy Nguyen-Tuong, Lehel Csató, Bernhard Schölkopf, and Jan Peters. Learning inverse kinematics with structured prediction. In *IEEE International Conference on Intelligent Robots and Systems*, pages 698–703, 2011.

[12] Georgios Boutsioukis, Ioannis Partalas, and Ioannis Vlahavas. *Transfer Learning in Multi-Agent Reinforcement Learning Domains*, pages 249–260. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[13] D. Brugali and P. Scandurra. Component-based robotic engineering (part i) [tutorial]. *IEEE Robotics Automation Magazine*, 16(4):84–96, December 2009.

[14] D. Brugali and A. Shakhimardanov. Component-based robotic engineering (part ii). *IEEE Robotics Automation Magazine*, 17(1):100–112, March 2010.

[15] H. Bruyninckx. Open robot control software: the orocos project. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 3, pages 2523–2528, 2001.

[16] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E. Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3352–3358. AAAI Press, 2015.

[17] S. Calinon and A. Billard. Active teaching in robot programming by demonstration. In *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 702–707, Aug 2007.

[18] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics Automation Magazine*, 17(2):44–54, June 2010.

[19] Sylvain Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intell. Serv. Robot.*, 9(1):1–29, January 2016.

[20] Raffaello Camoriano, Silvio Traversaro, Lorenzo Rosasco, Giorgio Metta, and Francesco Nori. Incremental semiparametric inverse dynamics learning. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 544–550, 2016.

[21] Kian Ming Chai, Christopher Williams, Stefan Klanke, and Sethu Vijayakumar. Multi-task Gaussian process learning of robot inverse dynamics. In *Nips*, 2009.

[22] J. Chen and H. Y. K. Lau. Learning the inverse kinematics of tendon-driven soft manipulators with k-nearest neighbors regression and gaussian mixture regression. In *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, pages 103–107, April 2016.

[23] J. Chen, H. Y. K. Lau, W. Xu, and H. Ren. Towards transferring skills to flexible surgical robots with programming by demonstration and reinforcement learning. In *2016 Eighth International Conference on Advanced Computational Intelligence (ICACI)*, pages 378–384, Feb 2016.

[24] Sonia Chernova and Manuela Veloso. Confidence-based multi-robot learning from demonstration. *International Journal of Social Robotics*, 2(2):195–215, Jun 2010.

[25] Younggeun Choi, Shin-Young Cheong, and Nicolas Schweighofer. Local Online Support Vector Regression for Learning Control. In *International Symposium on Computational Intelligence in Robotics and Automation*, number 2, pages 13–18, 2007.

[26] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba. Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model. *ArXiv e-prints*, October 2016.

[27] Gabriele Costante, Thomas A. Ciarfuglia, Paolo Valigi, and Elisa Ricci. Transferring knowledge across robots: A risk sensitive approach. *Robotics and Autonomous Systems*, 65(Supplement C):1 – 14, 2015.

[28] J. S. De La Cruz, W. Owen, and D. Kulić. Online Learning of Inverse Dynamics via Gaussian Process Regression. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, number 1, pages 3583–3590, 2012.

[29] Zhen Cui, Hong Chang, Shiguang Shan, and Xilin Chen. Generalized unsupervised manifold alignment. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2429–2437. Curran Associates, Inc., 2014.

[30] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 540–545. AAAI Press, 2007.

[31] B. Dariush, M. Gienger, A. Arumbakkam, C. Goerick, Youding Zhu, and K. Fujimura. Online and markerless motion retargeting with kinematic constraints. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 191–198, Sept 2008.

[32] Joseph Sun de la Cruz, Ergun Calisgan, Dana Kulić, William Owen, and Elizabeth A. Croft. On-line dynamic model learning for manipulator control. *IFAC Proceedings Volumes*, 45(22):869 – 874, 2012. 10th IFAC Symposium on Robot Control.

[33] Joseph Sun de la Cruz, Dana Kulić, and William Owen. A comparison of classical and learning controllers. *IFAC Proceedings Volumes*, 44(1):1102 – 1107, 2011. 18th IFAC World Congress.

[34] Joseph Sun de la Cruz, Dana Kulić, and William Owen. *Online Incremental Learning of Inverse Dynamics Incorporating Prior Knowledge*, pages 167–176. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[35] B. Delhaisse, D. Esteban, L. Rozo, and D. Caldwell. Transfer learning of shared latent spaces between robots with similar kinematic structure. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4142–4149, May 2017.

[36] Yiannis Demiris and Anthony Dearden. From motor babbling to hierarchical learning by imitation: a robot developmental pathway, 2005.

[37] Fernando Diaz and Donald Metzler. Pseudo-aligned multilingual corpora. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2727–2732, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[38] M. Do, P. Azad, T. Asfour, and R. Dillmann. Imitation of human motion on a humanoid robot using non-linear optimization. In *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, pages 545–552, Dec 2008.

[39] Alain Droniou, Serena Ivaldi, Vincent Padois, and Olivier Sigaud. Autonomous online learning of velocity kinematics on the icub: a comparative study. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3577–3582, Vilamoura, Portugal, October 2012.

[40] Aaron D'Souza, Sethu Vijayakumar, and Stefan Schaal. Learning inverse kinematics. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 298–303, 2001.

[41] Carl Henrik Ek. *Shared Gaussian process latent variable models*. PhD thesis, Oxford Brookes University, 2009.

[42] Roy Featherstone. *Rigid Body Dynamics Algorithms*, section 3.1, pages 39–64. Springer, New York, 2008.

[43] Yin Feng, Wang Yao-nan, and Yang Yi-min. Inverse kinematics solution for robot manipulator based on neural network under joint subspace. *International Journal of Computers Communications & Control*, 7(3):459–472, 2014.

[44] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[45] M. Field, D. Stirling, Z. Pan, and F. Naghdy. Learning trajectories for robot programing by demonstration using a coordinated mixture of factor analyzers. *IEEE Transactions on Cybernetics*, 46(3):706–717, March 2016.

[46] S. Forestier, Y. Mollard, and P.-Y. Oudeyer. Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning. *ArXiv e-prints*, August 2017.

[47] Stefan Gärtner, Martin Do, Tamim Asfour, Rüdiger Dillmann, Christian Simonidis, and Wolfgang Seemann. Generation of human-like motion for humanoid robots based on marker-based motion capture data. In *ISR/ROBOTIK*, pages 1–8. VDE Verlag, 2010.

[48] Arjan Gijsberts and Giorgio Metta. Incremental learning of robot dynamics using random features. In *2011 IEEE International Conference on Robotics and Automation*, pages 951–956, 2011.

[49] Arjan Gijsberts and Giorgio Metta. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. *Neural Networks*, 41:59–69, 2013.

[50] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, June 2012.

[51] R. Gopalan, Ruonan Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *2011 International Conference on Computer Vision*, pages 999–1006, Nov 2011.

[52] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2625–2633. Curran Associates, Inc., 2013.

[53] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning. In *Proceedings - IEEE International Conference on Learning Representations*, 2017.

[54] Mohamed K. Helwa and Angela Schoellig. Multi-Robot Transfer Learning: A Dynamical System Perspective. Under review at International Conference on Intelligent Robots and Systems (IROS) 2017, May 2017.

[55] Jorge David Figueroa Heredia, Jose Ildefonso U. Rubrico, Shouhei Shirafuji, and Jun Ota. Teaching tasks to multiple small robots by classifying and splitting a human example. *Journal of Robotics and Mechatronics*, 29(2):419–433, 2017.

[56] M. Hiratsuka, N. Makondo, B. Rosman, and O. Hasegawa. Trajectory learning from human demonstrations via manifold mapping. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3935–3940, Oct 2016.

[57] Tatsuya Hirose and Tadahiro Taniguchi. Abstraction multimodal low-dimensional representation from high-dimensional posture information and visual images. *Journal of Robotics and Mechatronics*, 25(1):80–88, 2013.

[58] Jacob Huckaby and Henrik Christensen. A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics. In *AAAI Workshops*, 2012.

[59] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Comput.*, 25(2):328–373, February 2013.

[60] Hyunryong Jung, Arjun Menon, and Ronald C. Arkin. A conceptual space architecture for widely heterogeneous robotic systems. In *Proc. 2nd International Conference on Biologically Inspired Cognitive Architectures (BICA 2011*, 2011.

[61] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, Oct 2011.

[62] Z. Kira. Transferring embodied concepts between perceptually heterogeneous robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4650–4656, Oct 2009.

[63] Z. Kira. Inter-robot transfer learning for perceptual classification. In *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2010*, 2010.

[64] Z. Kira. Using conceptual spaces to fuse knowledge from heterogeneous robot platforms. In *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2010*, volume 7710 of *procspie*, April 2010.

[65] Z. Kira. Transfer of sparse coding representations and object classifiers across heterogeneous robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2209–2215, Sept 2014.

[66] Z. Kira. An evaluation of features for classifier transfer during target handoff across aerial and ground robots. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4245–4251, May 2015.

[67] Zsolt Kira. Mapping grounded object properties across perceptually heterogeneous embodiments. In *22nd International FLAIRS Conference*, pages 57–62, 2009.

[68] Jens Kober and Jan R. Peters. Policy search for motor primitives in robotics. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 849–856. Curran Associates, Inc., 2009.

[69] J. Koenemann, F. Burget, and M. Bennewitz. Real-time imitation of human whole-body motions by humanoids. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2812, May 2014.

[70] RaşIt KöKer. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Inf. Sci.*, 222:528–543, February 2013.

[71] Raşit Köker, Tarik Çakar, and Yavuz Sari. A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators. *Eng. with Comput.*, 30(4):641–649, October 2014.

[72] Hitoshi Kono, Akiya Kamimura, Kohji Tomita, and Tsuyoshi Suzuki. Transfer learning method using ontology for heterogeneous multi-agent reinforcement learning. *International Journal of Advanced Computer Science & Applications*, 5(10), 2014.

[73] Alessandro Lazaric. *Transfer in Reinforcement Learning: A Framework and a Survey*, pages 143–173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[74] J. Leitner, S. Harding, M. Frank, A. Frster, and J. Schmidhuber. Transferring spatial perception between robots operating in a shared workspace. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1507–1512, Oct 2012.

[75] G. H. Lim, I. H. Suh, and H. Suh. Ontology-based unified robot knowledge for service robots in indoor environments. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(3):492–509, May 2011.

[76] H. Liu and P. Singh. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, October 2004.

[77] Robert Loftin, Bei Peng, James MacGlashan, Michael L. Littman, Matthew E. Taylor, Jeff Huang, and David L. Roberts. Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, 30(1):30–59, Jan 2016.

[78] Jie Luo, Andrzej Pronobis, and Barbara Caputo. SVM-based transfer of visual knowledge across robotic platforms. In *Proceedings of the 5th International Conference on Computer Vision Systems (ICVS'07)*, Bielefeld, Germany, March 2007.

[79] G. Maeda, M. Ewerton, D. Koert, and J. Peters. Acquiring and generalizing the embodiment mapping from human observations to robot skills. *IEEE Robotics and Automation Letters*, 1(2):784–791, July 2016.

[80] N. Makondo, J. Claassens, N. Tlale, and M. Braae. Geometric technique for the kinematic modeling of a 5 dof redundant manipulator. In *2012 5th Robotics and Mechatronics Conference of South Africa*, pages 1–7, Nov 2012.

[81] Ndivhuwo Makondo, Benjamin Rosman, and Osamu Hasegawa. Knowledge Transfer for Learning Robot Models via Local Procrustes Analysis. In *15th IEEE-RAS International Conference on Humanoid Robotics*, Seoul, 2015. IEEE.

[82] Christian Mandery, Ömer Terlemez, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. Unifying representations and large-scale whole-body motion databases for studying human motion. *IEEE Trans. Robotics*, 32(4):796–809, 2016.

[83] J. Matai, Y. H. Suh, H. Kim, K. W. Lee, and H. Kim. Integration framework for interoperability of distributed and heterogeneous robot middlewares. In *2008 10th International Conference on Control, Automation, Robotics and Vision*, pages 2337–2343, Dec 2008.

[84] Franziska Meier, Philipp Hennig, and Stefan Schaal. Efficient Bayesian local model learning for control. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2244–2249, 2014.

[85] Franziska Meier, Philipp Hennig, and Stefan Schaal. Incremental Local Gaussian Regression. *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, (M):972–980, 2014.

[86] Franziska Meier, Daniel Kappler, Nathan Ratliff, and Stefan Schaal. Towards Robust Online Inverse Dynamics Learning. In *International Conference on Intelligent Robots and Systems*, pages 4034–4039, 2016.

[87] Franziska Meier and Stefan Schaal. Drifting Gaussian Processes with Varying Neighborhood Sizes for Online Model Learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, 2016. IEEE.

[88] Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco Nori. The icub humanoid robot: An open platform for research in embodied cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages 50–56, New York, NY, USA, 2008. ACM.

[89] C. Moulin-Frier and P. Y. Oudeyer. Exploration strategies in developmental robotics: A unified probabilistic framework. In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–6, Aug 2013.

[90] Thibaut Munzer, Freek Stulp, and Olivier Sigaud. Non-linear regression algorithms for motor skill acquisition: a comparison. In *9èmes Journées Francophones de Planification, Décision et Apprentissage*, Liège, Belgium, May 2014.

[91] Chrystopher L. Nehaniv and Kerstin Dautenhahn. Like me?- measures of correspondence and imitation. *Cybernetics and Systems*, 32(1-2):11–51, 2001.

[92] Sao Mai Nguyen and Pierre-Yves Oudeyer. Socially guided intrinsic motivation for robot learning of motor skills. *Autonomous Robots*, 36(3):273–294, Mar 2014.

[93] Duy Nguyen-Tuong and Jan Peters. Incremental Sparsification for Real-time Online Model Learning. In *Artificial Intelligence and Statistics*, volume 9, pages 557–564, 2010.

[94] Duy Nguyen-tuong and Jan Peters. Using model knowledge for learning inverse dynamics. In *2010 IEEE International Conference on Robotics and Automation*, pages 2677–2682, 2010.

[95] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: A survey. *Cognitive Processing*, 12(4):319–340, 2011.

[96] Duy Nguyen-Tuong, Jan Peters, Matthias Seeger, and Bernhard Schölkopf. Learning Inverse Dynamics : a Comparison. In *Advances in Computational Intelligence and Learning: Proceedings of the European Symposium on Artificial Neural Networks (ESANN 2008)*, pages 13–18, 2008.

[97] Duy Nguyen-Tuong, Bernhard Scholkopf, and Jan Peters. Sparse online model learning for robot control with support vector regression. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, number 1, pages 3121–3126, 2009.

[98] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Model Learning with Local Gaussian Process Regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[99] E. Oyama, T. Maeda, J. Q. Gan, E. M. Rosales, K. F. MacDorman, S. Tachi, and A. Agah. Inverse kinematics learning for robotic arms with fewer degrees of freedom by modular neural network systems. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1791–1798, Aug 2005.

[100] A. Paikan, D. Schiebener, M. Wchter, T. Asfour, G. Metta, and L. Natale. Transferring object grasping knowledge and skill across different robotic platforms. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 498–503, July 2015.

[101] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[102] Sinno Jialin Pan, James T. Kwok, and Qiang Yang. Transfer learning via dimensionality reduction. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, pages 677–682. AAAI Press, 2008.

[103] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1187–1192, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[104] J. Peters and D. Nguyen-Tuong. Real-time learning of resolved velocity control on a mitsubishi pa-10. In *2008 IEEE International Conference on Robotics and Automation*, pages 2872–2877, May 2008.

[105] J. Peters and S. Schaal. Learning to Control in Operational Space. *The International Journal of Robotics Research*, 27(2):197–212, 2008.

[106] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, May 2008.

[107] A. S. Polydoros and L. Nalpantidis. A reservoir computing approach for learning forward dynamics of industrial manipulators. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 612–618, Oct 2016.

[108] Athanasios S. Polydoros, Lazaros Nalpantidis, and Volker Kruger. Real-time deep learning of robotic manipulator inverse dynamics. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3442–3448, 2015.

[109] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[110] Kaizad Raimalwala, Bruce Francis, and Angela Schoellig. A Preliminary Study of Transfer Learning between Unicycle Robots. In *AAAI Spring Symposium*, pages 53–59, 2016.

[111] Kaizad V. Raimalwala, Bruce A. Francis, and Angela P. Schoellig. An upper bound on the error of alignment-based Transfer Learning between two linear, time-invariant, scalar systems. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5253–5258, 2015.

[112] Nathan Ratliff, Franziska Meier, Daniel Kappler, and Stefan Schaal. Doomed: Direct online optimization of modeling errors in dynamics. *Big Data*, 4(4):253–268, 2016.

[113] E. Rohmer, S. P. N. Singh, and M. Freese. V-rep: a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[114] M. Rolf, J. J. Steil, and M. Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3):216–229, Sept 2010.

[115] M. Rolf, J. J. Steil, and M. Gienger. Online goal babbling for rapid bootstrapping of inverse models in high dimensions. In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pages 1–8, Aug 2011.

[116] Diego Romeres, Mattia Zorzi, Raffaello Camoriano, and Alessandro Chiuso. Online semi-parametric learning for inverse dynamics modeling. In *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, pages 2945–2950, 2016.

[117] B. Rosman and S. Ramamoorthy. What good are actions? accelerating learning using learned action priors. In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–6, Nov 2012.

[118] B. Rosman and S. Ramamoorthy. Action priors for learning domain invariances. *IEEE Transactions on Autonomous Mental Development*, 7(2):107–118, June 2015.

[119] Steve Rowe and Christopher R Wagner. An introduction to the joint architecture for unmanned systems (jaus). *Ann Arbor*, 1001:48108, 2008.

[120] R. Saegusa, G. Metta, G. Sandini, and S. Sakka. Active motor babbling for sensorimotor learning. In *2008 IEEE International Conference on Robotics and Biomimetics*, pages 794–799, Feb 2009.

[121] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. Kramer, and E. Miguelez. An ieee standard ontology for robotics and automation. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1337–1342, Oct 2012.

[122] Zeeshan Shareef, Felix Reinhart, and Jochen J Steil. Generalizing the Inverse Dynamic Model of KUKA LWR IV+ for Load Variations using Regression in the Model Space. In *Proceedings of IEEE Int. Conf. Intelligent Robots and Systems*, 2016.

[123] Jie Sheng, Sam Chung, Leo Hansel, Don McLane, Joel Morrah, Seung-Ho Baeg, and Sangdeok Park. Jaus to ethercat bridge: Toward real-time and deterministic joint architecture for unmanned systems. *J. Control Sci. Eng.*, 2014:1:1–1:1, January 2014.

[124] A. P. Shon, K. Grochow, and R. P. N. Rao. Robotic imitation from human motion capture using gaussian processes. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pages 129–134, Dec 2005.

[125] Aaron Shon, Keith Grochow, Aaron Hertzmann, and Rajesh P Rao. Learning shared latent structure for image synthesis and robotic imitation. In Y. Weiss, P. B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1233–1240. MIT Press, 2006.

[126] W. D. Smart and L. Pack Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3404–3410 vol.4, 2002.

[127] Mark W Spong, Seth Hutchison, and M Vidyasagar. *Robot modeling and control*, chapter 1.2. Addison-Wesley, 2006.

[128] Christopher Stanton, Anton Bogdanovych, and Edward Ratanasena. Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning. In *Proceedings of Australasian Conference on Robotics and Automation (ACRA)*, 2012.

[129] Freek Stulp and Olivier Sigaud. Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn. Journal of Behavioral Robotics*, 4(1):49–61, September 2013.

[130] Jürgen Sturm, Christian Plagemann, and Wolfram Burgard. Body schema learning for robotic manipulators from visual self-perception. *Journal of Physiology-Paris*, 103(3):220–231, 2009.

[131] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[132] Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Trans. Graph.*, 24(1):98–117, January 2005.

[133] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, December 2009.

[134] Matthew E Taylor and Peter Stone. An introduction to intertask transfer for reinforcement learning. *Ai Magazine*, 32(1):15, 2011.

[135] Matthew E. Taylor, Halit Bener Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, pages 617–624, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[136] Moritz Tenorth and Michael Beetz. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research*, 32(5):566–590, 2013.

[137] Moritz Tenorth, Alexander Clifford Perzylo, Reinhard Lafrenz, and Michael Beetz. The roboearth language: Representing and exchanging knowledge about actions, objects, and environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1284–1289. IEEE, 2012.

[138] Moritz Tenorth, Alexander Clifford Perzylo, Reinhard Lafrenz, and Michael Beetz. Representation and exchange of knowledge about actions, objects, and environments in the roboearth framework. *IEEE Transactions on Automation Science and Engineering*, 10(3):643–651, 2013.

[139] Ö. Terlemez, S. Ulbrich, C. Mandery, M. Do, N. Vahrenkamp, and T. Asfour. Master motor map (mmm) – framework and toolkit for capturing, representing, and reproducing human motion on humanoid robots. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 894–901, Nov 2014.

[140] Jo-Anne Ting, Franziska Meier, Sethu Vijayakumar, and Stefan Schaal. *Locally Weighted Regression for Control*, pages 1–14. Springer US, Boston, MA, 2016.

[141] Tarik Tosun, Ross Mead, and Robert Stengel. A general method for kinematic retargeting: Adapting poses between humans and robots. In *ASME 2014 International Mechanical Engineering Congress and Exposition*, 2014.

[142] N. G. Tsagarakis, S. Morfey, G. Medrano Cerda, L. Zhibin, and D. G. Caldwell. Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control. In *2013 IEEE International Conference on Robotics and Automation*, pages 673–678, May 2013.

[143] A. Ude, B. Nemec, T. Petri, and J. Morimoto. Orientation in cartesian space dynamic movement primitives. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2997–3004, May 2014.

[144] Terry Taewoong Um, Myoung Soo Park, and Jung-min Park. Independent Joint Learning : A Novel Task - to - Task Transfer Learning Scheme for Robot Models. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 5679–5684, 2014.

[145] J. Verbeek. Learning nonlinear image manifolds by global alignment of local linear models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1236–1250, Aug 2006.

[146] Sethu Vijayakumar, Aaron D'Souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural computation*, 17(12):2602–2634, 2005.

[147] Markus Waibel, Michael Beetz, Javier Civera, Raffaello d'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Häussermann, Rob Janssen, JMM Montiel, Alexander Perzylo, et al. Roboearth. *IEEE Robotics & Automation Magazine*, 18(2):69–82, 2011.

[148] Chang Wang and Sridhar Mahadevan. Manifold alignment using Procrustes analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1120–1127, Helsinki, 2008.

[149] Chang Wang and Sridhar Mahadevan. Manifold alignment without correspondence. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 1273–1278, 2009.

[150] Chang Wang and Sridhar Mahadevan. Manifold alignment preserving global geometry. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1743–1749. AAAI Press, 2013.

[151] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, May 2016.

[152] Dit Yan Yeung and Yu Zhang. Learning inverse dynamics by Gaussian process regression under the multi-task learning framework. *The Path to Autonomous Robots: Essays in Honor of George A. Bekey*, pages 131–142, 2009.

[153] Deming Zhai, Bo Li, Hong Chang, Shiguang Shan, Xilin Chen, and Wen Gao. Manifold alignment via corresponding projections. In *British Machine Vision Conference, BMVC 2010, Aberystwyth, UK, August 31 - September 3, 2010. Proceedings*, pages 1–11, 2010.

[154] Yusen Zhan and Matthew E. Taylor. Online Transfer Learning in Reinforcement Learning Domains. In *Proceedings of the AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (SDMIA)*, November 2015.

[155] L. Zhang, Z. Cheng, Y. Gan, G. Zhu, P. Shen, and J. Song. Fast human whole body motion imitation algorithm for humanoid robots. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1430–1435, Dec 2016.