Skill Transfer between Industrial Robots by Sparse learning

By

Ke Lan

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

May, 2017

Nashville, Tennessee

Approved:

Richard Alan Peters, Ph.D.

D. Mitchell Wilkes, Ph.D.

In dedication to my advisor and partner for supporting me all the way

ACKNOWLEDGMENTS

The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

TABLE OF CONTENTS

Page

DEDICATION ii

ACKNOWLEDGMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

Chapter

1 INTRODUCTION 1

1.1 Related Work 3

1.1.1 Learning by demonstration and knowledge transfer across robots 3

1.1.2 Compressed sensing 4

2 MOTION CAPTURE AND DEFINITION 5

2.1 Kinect Motion Capture 5

2.2 ASF/AMC Motion System 7

2.2.1 ASF file 7

2.2.2 AMC file 9

3 SKILL TRANSFER SYSTEM 10

3.1 Potential Motion Trajectory Generation 10

3.2 Motion Registration between Robots 14

3.3 Dynamic Movement Primitives Learning 18

3.4 DMP learning and Motion representation 23

3.5 Skill transfer and inverse kinematics 25

4 SYSTEM DESCRIPTION 27

4.1 CMU Motion Database 27

4.2 Yaskawa Motoman HP3JC 28

4.3 Rethink Robotics Baxter 30
5 EXPERIMENTS AND ANALYSIS 33
5.1 Potential Motion Trajectory Generation 33
5.2 Motion Registration between Robots 34
5.3 Dynamic Movement Primitives Learning 35
5.4 DMP learning and Motion representation 40
5.5 Skill transfer and advantage in space complexity 40
6 CONCLUSION 44
BIBLIOGRAPHY 45

LIST OF TABLES

Table Page

4.1 HP3JC joints constraints 30

4.2 Baxter joints constraints 30

LIST OF FIGURES

Figure Page

2.1 Kinect and different cameras[1] 6

2.2 Kinect depth image and corresponding skeleton generated from SDK 6

2.3 ASF hierarchy in CMU Motion database 8

3.1 Pipeline of skill transfer system[2] 11

3.2 Linkage structure between two joints[3] 13

3.3 Standard Motions registration result between human and Baxtor. In this figure, the corresponding motion is straight right arm sweep. 17

3.4 Structure of Neural Sparse Autoencoder 20

3.5 the sigmoid function 21

3.6 Canonical system in DMP algorithm 24

3.7 Kernel activation in DMP algorithm 25

4.1 Front and back view of marker set in CMU Motion Database[4] 28

4.2 Yaskawa Motoman HP3JC[5] 29

4.3 Side view of Yaskawa Motoman HP3JC[6] 29

4.4 Rethink Robotics Baxter[7] 31

4.5 Front and side view of Baxter[8] 32

5.1 Data from CMU motion database. Top: Boxing. Bottom: Washing window. 33

5.2 Motion data generated by our method 34

5.3 Pre-registration and resultant point cloud between human and Baxter 36

5.4 Visualization of sparse coding result. The original data is represented by the value of pixel. 37

5.5 25 learned bases of end-effector trajectories 38

5.6 Process of solving Lasso [1] 39
5.7 Original trajectory (red) and linear combination fitting result (blue) from bases 39
5.8 Error distribution of dictionary representation over 10,000 motion trajectories 40
5.9 DMP of a movement base 41
5.10 Skill transfer between human and Baxter, Baxter and Motoman 41
5.11 Difference of space complexity between direct transfer and our method . . . 42
5.12 Difference of space requirement between direct transfer and our method . . 43

Chapter 1

INTRODUCTION

Robots unconsciously but drastically changed the lives of many individuals. In 1956, inspired by the short stories and novels of Isaac Asimov, Devol and Engelberger brainstormed to derive the first industrial robot arm, based upon Devols patent, called the Unimate. Programmed Article Transfer became the seminal industrial robot patent which was ultimately sub-licensed around the world. [2]After development over decades, through we have not yet observed humanoid robot's acting as personal assistants in daily life, industrial robots are revolutionizing manufacturing. Assembled with cyber-physical systems, the Internet of things and cloud computing, industrial robots that can produce more intelligent and flexible products have become the bedrock of modern industry.

Recently, Boston Dynamics released its latest robot which is described the combination of legs and wheels as the best of both worlds. The wheels make Handle energy efficient on flat surfaces. With legs it's able to manage uneven terrain, and go nearly anywhere. [3] However, compared with the artificial intelligence being applied in research robotics, the operation technology of industrial robots is similar to 1970s when Stanford University produced the Stanford Cart which is designed to be a line follower but also was able to be controlled from a computer via radio link. [4]

Most industrial robots are still programmed by a human operator. Specific tasks are hard coded manually through a controller. Thus, task motions are preprogrammed and fixed point-by-point. The inability to modify pre-learned tasks during operation and the inability to transfer programs between different robots have limited the use of robots in industry. More sophisticated works such as cooperation between different robots and reproduction of complex skills from pre-learned knowledge are impossible without more intelligent robotic operation technology.

Furthermore, consider storage for robot skills. When a robot executes complex tasks, it must maintain a large library to store the redundant 3D position trajectories. To further the development of robots, it's necessary to generalize and transfer robotic skills between different robot platforms and to keep the motion library small and concise.

In this paper, a motion capture system, motion registration method, and motion dictionary, comprise a skill transfer system based on a sparse coding algorithm and the DMP method to address above two requirements. A raw skill is learned by capturing the 3D position of joints during human demonstration. To transfer the motion between different platforms, an initialization-enhanced self-adapted Scale-ICP method was designed to find a registration matrix between two robots. To embody the effect of motion compression and eliminate noise in original track, a motion dictionary was learned from potential trajectories generated from forward kinematics. Original motions are simplified as the linear combination of motion bases in the dictionary. Thus, over 10-k records of position can be compressed as a weighting vector with a low-dimension. Whenever we need to transfer or implement skills on the platform, the motion records are reconstructed by combing a weighting vector, a registration matrix, and a motion dictionary. With an inverse kinematics algorithm, to make the robot move, a resultant track is re-coded as operation commands for the specific robot.

Preliminary work [5] was enhanced with:

- An improved motion capture system which is capable of capturing the location of the full skeleton.
- A Scale ICP-based algorithm was designed to align two robots automatically.
- A dictionary learning algorithm was proposed to redefine and compress the original motion records.

1.1 Related Work

There are two main fields related to our work:

- Learning by demonstration generalizes skills by fitting a dynamical system model where the starting point and goal position of the task can be modified to keep the original behavior.

- Compressive sensing recovers a signal from far fewer samples. It exploits the sparsity of signals. Compressed sensing is one available method to generate primitives automatically and to decrease size of complex skills in the motion library.

1.1.1 Learning by demonstration and knowledge transfer across robots

In order to find a solution to reduce the motion programming time and reuse data from an older robot, researchers have been working on knowledge transfer across different robots for some time.

In recent years, researchers worked on the robot skill learning from human demonstration. Specifically, in Pastor's work [6], the observed movement is learned and represented by a non-linear differential system. This is called Dynamical Movement Primitives(DMP). In the DMP system, a learned skill can be reproduced with different starting and goal parameters to a required position. With the DMP method, different control models were built. Li [5] designed a Kinect-based skill transfer system. By applying a hierarchical reinforcement learning approach in sequences of Dynamic Movement Primitives, Stulp [7] proposed a method to find optimized parameters for DMP. Rückert [8] proposed an extension of dynamic movement primitives. By employing parametrized basis functions, it combines the benefits of DMP with muscle synergies.

To decompose complex trajectories into primitives, some researchers tried to represent task and robot knowledge as symbols. Konidaris [9] designed an algorithm that decomposed the complex task into a chain of component skills. Abbas [10] proposed a longest

common subsequence(LCS) algorithm to help build a generalized task structure.

1.1.2 Compressed sensing

Recently, compressive sensing has proliferated in the different fields of Computer Science and Electrical Engineering. By representing signals using only a few sparse weighting vector in a dictionary, compressive sensing (sparse approximation) exploits signal sparsity and compressibility. [11] After Bruckstein's work proved that most data such as images and signals can be recovered from a learned basis set, [12] researchers in different fields have implemented sparse approximation to compress the original signals. For instance, in image visual search, to compress the data transfered between client and server, Tibshirani [1] and Zou's [13] applied different sparse coding schemes to learn the dictionary and to compress the data from original BoW (Bag of Words) codebook.

The remainder of this thesis is organized as follows. In Chapter 2, motion capture using Microsoft Kinect and a skill representation system is briefly introduced. Chapter 3 presents the pipeline of the proposed skill transfer system which includes potential motion trajectory generation, motion registration, dynamical primitive motion codebook building, and target robot operation. In Chapter 4, the robot platforms used in experiment are described in detail. Experiments and discussions are given in Chapter 5 and conclusions are drawn in Chapter 6.

Chapter 2

MOTION CAPTURE AND DEFINITION

The motion capture and definition chapter describes the motion recording system of our project. First, using a Microsoft Kinect, human skills were recorded as a sequence of motions. After processing the raw joint data by low-pass filtering and normalization, the original human motion is stored in an ASF/AMC system [14], which provides a reasonable description of human and robot skeletal structure. Furthermore, based on the ASF/AMC system, the potential transformation of a skeleton is predictable. Thus, we can generate basic dynamical movement primitives from the skeletal system and decompose input motions into combinations of dynamical movement primitives for the convenience of future transfer between different robots.

2.1 Kinect Motion Capture

There are two independent different camera systems in the Kinect. A regular camera captures an RGB image of the scene. To obtain depth data, a depth sensor system in Kinect comprises an IR Projector and an IR Camera. The sensor in the camera is sensitive to the near-IR light from the IR Projector. A grid of generated lights is distorted by surfaces and imaged by the IR camera. By analyzing the deformations of the IR images of the scene, depth data are reconstructed. These three cameras restore the scene in 6 dimensions(color: R G B, position: X Y Z) [15].

To recognize and discriminate human body parts from the background, a machine learning based object recognition approach is implemented in the Kinect SDK. From a ground-truth database with 100,000 depth images, a randomized decision forest is trained which can map the pixel in depth image to corresponding human body parts. [16]

[1] https://www.codeproject.com/Articles/317974/KinectDepthSmoothing

Figure 2.1: Kinect and different cameras[1]

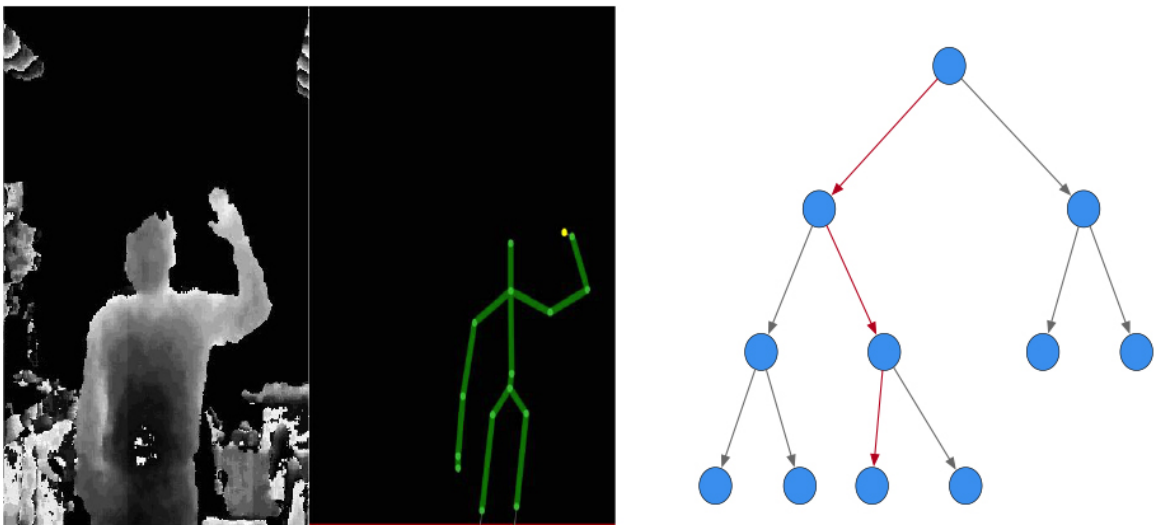

Figure 2.2: Kinect depth image and corresponding skeleton generated from SDK

Figure 2.2 demonstrates how a random tree decides which part of the human body the pixel belongs to. The randomized decision tree is learned from the ground-truth database. Each node in the tree represents condition which can divide the inputed data into two sets. Starting from the root of the tree, the pixel is tested in each node with a specific condition. Through a path in the tree (red edges in Figure 2.2), going down to a leaf node of the tree, the pixel is assigned to a specific body part.

After body part recognition by implementing the random forest, it's possible to estimate the joint positions by accumulating the 3D center of probability mass. The Mean Shift Algorithm is one method to find the modes in the density for each cluster. Motion is defined as the joint's position transformation in the time-line after low-pass filtering smoothing and normalization.

2.2 ASF/AMC Motion System

Human Motion capture as described above infers per-joint 3D position in the timeline. To represent captured motion from the joints' 3D position changes as a skeleton transformation, the information must be constrained by a reasonable skeleton model. A skeleton model is also necessary to generate proper primitive motions for skill transfer in the next step. In this project, the mechanical structure of human and robots is described by an Acclaim ASF file. Motion is restored in a corresponding AMC file [14].

2.2.1 ASF file

In the ASF file (Acclaim Skeleton File), the base pose of a skeleton is defined in a hierarchical style. Starting from the 3D position of a virtual joint root, the ASF file maintains the mechanical structure as a list of nodes. For each node, the specific description includes:

- "name" & "ID": "name" & "ID" provide a unique identification for the joint, which is used as reference for the hierarchy section.

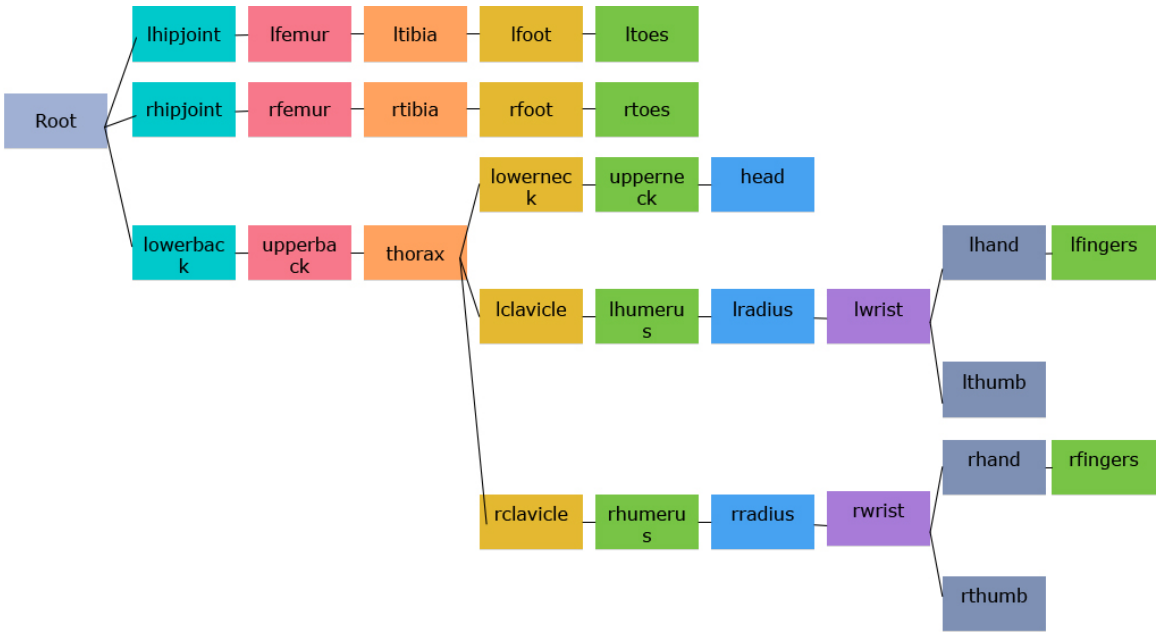

Figure 2.3: ASF hierarchy in CMU Motion database

- "parent": in ASF file, the relationship between nodes is described by the "parent" property in the hierarchy structure. For all nodes expect for the root, it keeps the parent node. Figure 2.3 is a classical ASF structure used in the CMU motion database to represent the human skeleton. Specific local properties can be obtained by recursive calculation from root to node.

- "direction" & "length" : "direction" and "length" define how the joint is posed with respect to the parent joint. "direction" provides default pose for the joint. "length" gives the distance to the joint from the parent.

- "axis", "dof" & "limits": "axis" provides the offset of global default axis. This property allows a joint to be drawn without all the information from the root. Since most joints have limitations in channels, "dof" gives the degrees of freedom for joint. For each available channel, "limits" defines the minimum and maximum of reasonable value.

2.2.2 AMC file

The AMC is used to record the transformation of joints in the timeline. For a specific node, based on the axis defined above, a rotation matrix Q is created in the order "XYZ":

$$Q(\theta_x, \theta_y, \theta_z) = Q_x(\theta_x)Q_y(\theta_y)Q_z(\theta_z) \tag{2.1}$$

Since the rotation matrix is orthogonal, the inverse matrix Q_{inv} is simply the transpose of the original Q.

In AMC file, each joint also maintains its axis translation offset from the parent root. A translation matrix T can be created from this information. In the same way, a motion transformation matrix M can be obtained for each channel. These are combined in a pre-calculated matrix. The local transformation is, therefore, defined as:

$$L = Q_{inv}MQT \tag{2.2}$$

Since the hierarchical relationship is defined in the ASF file, a global transformation can be fixed by accumulation of the local transformations:

$$G = L_1 L_2 \ldots L_{root} \tag{2.3}$$

Chapter 3

SKILL TRANSFER SYSTEM

This chapter describes the skill transfer system in detail. As illustrated in Figure 3.1, the skill transfer system consists of an offline part and an online part. To learn a dictionary which can provide sparse approximation for potential movements in different robots, a training set, with a large number of trajectories, is randomly generated. This is done offline using the DH parameters of the specific robot. To evaluate the registration relation between two platforms, a set of standard movements is designed. Aligning the movement from source to target, combined with the potential trajectory data, the transformation between these two platform is revealed by applying the Scale-ICP algorithm [17]. Based on the generated trajectories, a motion dictionary is learned by a sparse auto-encoder [18] in dynamic movement primitives learning part. When the movement primitives are available, the original movement is recoded by solving a lasso problem [1] to find the optimal weighting vector in the resultant dictionary. For each movement primitive, the DMP algorithm is implemented to create a dynamical system for further skill generalization. In the online part, whenever a skill is to be transferred, only the weighting vector is necessary to transfer. The motion is recovered from a linear combination of primitives in the dictionary and registration via the transformation described above. By using inverse kinematics, the transfered skill is translated into platform-specific operation for the target platform.

3.1 Potential Motion Trajectory Generation

In this part, the potential motion trajectory is generated as training data for motion dictionary learning and trajectory registration between different models.

Based on the mechanical structure of the robot model, forward kinematics uses kinematic equation and corresponding parameters of each joint to computer the 3D position

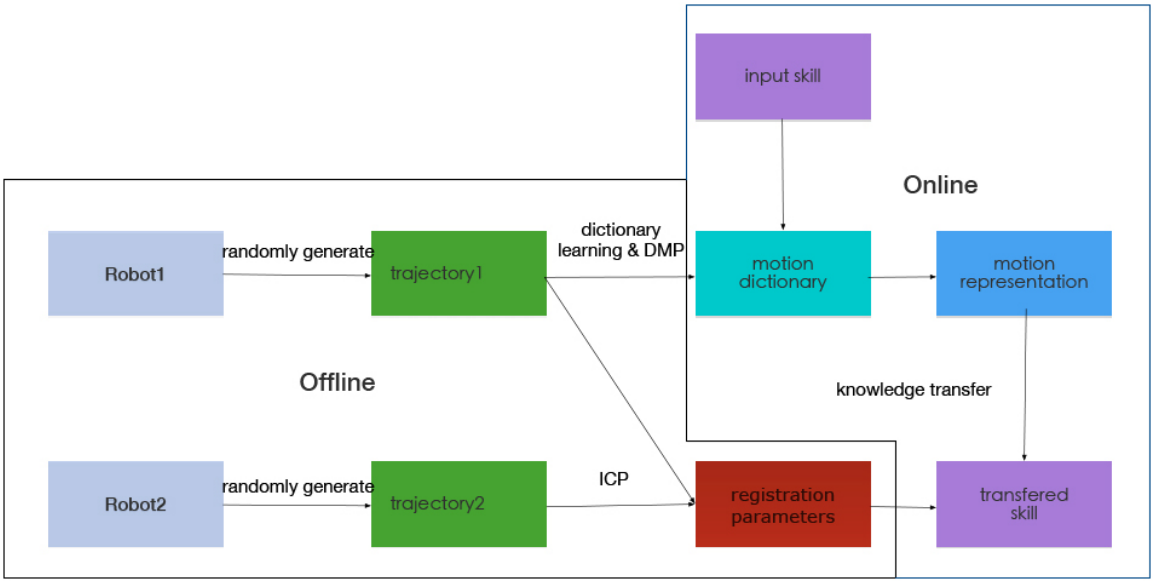

Figure 3.1: Pipeline of skill transfer system[1]

and pose for joints. Denavit Hartenberg parameters (DH parameters) is a well known and frequently used forward kinematics algorithm [19]. The DH parameters method describes the model as a chain of joints with linkages. For link i, the transformation matrix T_i is defined as the combination of four primitive transformations [20]:

$$T_i = Trans_{z_{i-1}}(d_i)Rot_{z_{i-1}}(\theta_i)Trans_{x_i}(a_i)Rot_{x_i}(\alpha_i) \tag{3.1}$$

where d_i is the link offset distance between local coordinate of joint $i-1$ to local coordinate of joint i in axes $z_i - 1$.

θ_i is the angle offset about previous link in x axes.

a_i is link length measurement for link i.

α_i is angle offset about previous link in z axes.

The four primitive transformation are respectively:

$$Trans_{z_{i-1}}(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$Rot_{z_{i-1}}(\theta_i) = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$Trans_{x_i}(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$Rot_{x_i}(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_i & -sin\alpha_i & 0 \\ 0 & sin\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

As a result, T_i equals to:

$$T_i = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

When the mechanical parameters are obtained, starting from the first link l_0, by implementing transformation $T_0 T_1 \dots T_{i-1}$ in chain, the l_i is fixed.

[2]http://www.intechopen.com/books/matlab-a-fundamental-tool-for-scientific-computing-and-

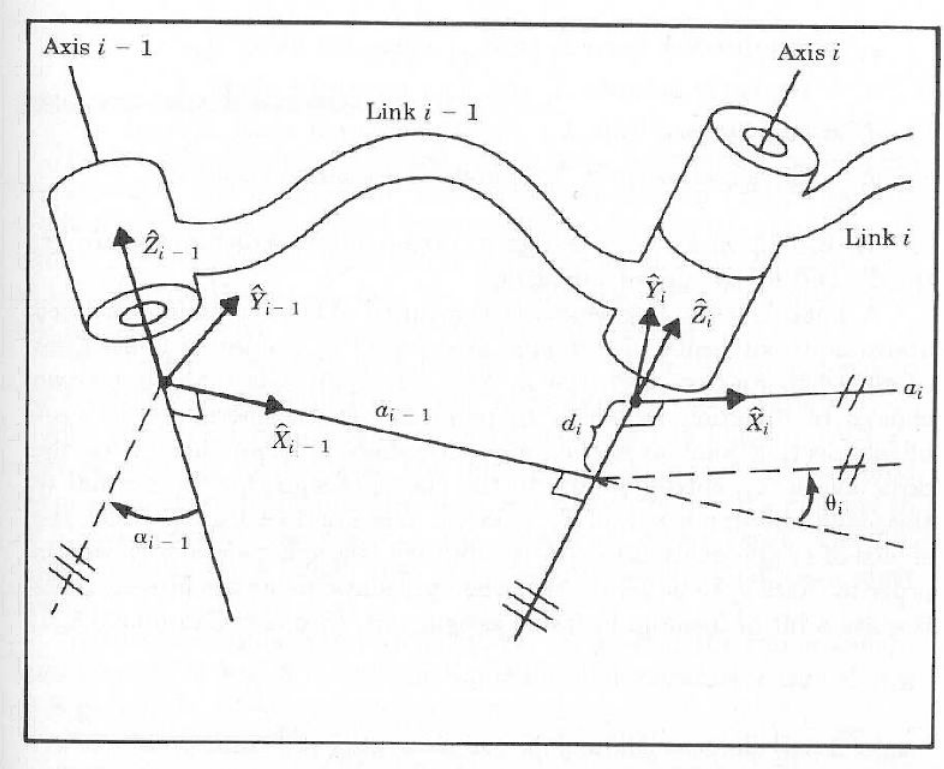

Figure 3.2: Linkage structure between two joints[2]

In our system, since the length of linkage is fixed, the system can be simplified as:

$$P = T_{1\ldots n-1}(DH parameters, \theta) \tag{3.7}$$

where P is the location of $joint_{1\ldots n}$, T is the transformation for $linkage_{1\ldots n-1}$, and θ is the set of input angle parameters for $joint_{1\ldots n}$.

To generate the training dataset, which can almost cover all the potential trajectory of motion, in our method for each joint four anchoring angles are generated randomly. After interpolation for these anchoring angles, we create a sequence of θ for all angles. Applying the DH Parameters described above, the full trajectory is produced. Consequently, in this way, we both guarantee the diversity of the training set and the smoothness of the motion track.

engineering-applications-volume-3/micro-robot-management

3.2 Motion Registration between Robots

In this section, to transfer skills between platforms, we are concerned with the registration of two trajectories from different robots. First, the registration problem is considered as a point registration optimization problem. Then, a scale factor is introduced into the optimization formula. Since the formula describes the registration error, by solving it, the registration can be solved uniformly across the links. Last, by applying the Scale-ICP algorithm from Ying's work [17], the original motion from one robot can be transformed into the corresponding motion in target robot.

Suppose, in the data generated by the forward kinematics described above, there are two partially overlapping trajectories, an original trajectory P, and a target robot motion Q. Denote $s, R \in \mathbb{R}^{3\times3}, \vec{t} \in \mathbb{R}^3$ as the scale factor, rotation matrix in 3D space, and translation vector. The registration optimization model can be defined as:

$$\begin{aligned} &\min_{s,R,\vec{t}}(\sum_{\vec{p}_a \in P} \|sR\vec{p}_a + \vec{t} - \vec{q}_{c(a)}\|_2^2) \\ &s.t. R^T R = I_3, det(R) = 1 \end{aligned} \tag{3.8}$$

where $\vec{q}_{c(a)}$ is the corresponding point for $\vec{p}_a$ in the target trajectory scan.

Suppose the correspondence of points between two trajectory sets is not available, in our work, compared with the standard ICP algorithm [21], the optimization is designed as a iterative process.

In iteration k, to map points in the original trajectory to the potential trajectory of the target robot, a closest matching strategy is applied at first. More specifically, a K-D tree [22] is constructed based on the data from Q. For $\vec{p}_a \in P_k$, nearest neighbors in the target data set Q are found to produce the corresponding closest data set Q_k.

The target function of the registration optimization problem can be redefined as a Fidu-

cial Registration Error [23] without the scale factor s:

$$FRE^2 = \frac{1}{N}\sum_{i=1}^{N} \|Rp_i + \vec{t} - q_i\|^2 \tag{3.9}$$

In this equation, to separate translation vector $\vec{t}$, standard FRE is processed by local mean subtraction as:

$$FRE^2 = \frac{1}{N}\sum_{i=1}^{N} \|R\widetilde{p_i} - \widetilde{q_i}\|^2 + \|R\bar{p_i} - \bar{q_i} + \vec{t}\|^2 \tag{3.10}$$

where $\bar{p_i}$ and $\bar{q_i}$ are the mean values of point clouds. $\widetilde{p_i}$ and $\widetilde{q_i}$ are corresponding point location generated from mean subtraction. As we know, R and $\vec{t}$ represent the rigid transformation between the original points and the target points. This registration relation is also valid to the mean values. Thus, we have $\|R\bar{p_i} - \bar{q_i} + \vec{t}\|^2 = 0$. Consequently, the translation vector can be calculated by $\vec{t} = \bar{q_i} - R\bar{p_i}$.

The rest of Fiducial Registration Error is:

$$\frac{1}{N}\sum_{i=1}^{N} \|R\widetilde{p_i} - \widetilde{q_i}\|^2 = \frac{1}{N} trace(Rp_i - q_i)(Rp_i - q_i)^t \tag{3.11}$$

Let $H = pq^t$, by using singular-value decomposition of H to find parameter that maximizes trace of RH, rotation matrix R can be obtained:

$$H = U \wedge V \tag{3.12}$$

$$R = \begin{cases} VU^T det(VU^T) = 1 \\ V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} U^T, det(VU^T) = -1 \end{cases} \tag{3.13}$$

when rotation matrix R is obtained, to extend this algorithm, a scale factor is introduced

in the registration problem as described in Scale-ICP algorithm [17],

$$s = \frac{\sum_{i=1}^{N} < R\widetilde{p}_i, \widetilde{q}_i >}{s} \tag{3.14}$$

After we get the scale factor s_k and rotation matrix R_k, in current iteration, the corresponding translation vector separated in equation 3.10 is generated by:

$$\vec{t} = \bar{q} - s_k R_k \bar{p} \tag{3.15}$$

Through the Scale-ICP algorithm, the original motion from one robot can be transformed into the corresponding motion for the target robot. However, since ICP algorithm is sensitive to initial parameters, bad initialization often leads to convergence to a local minimum which can be incorrect. We designed a set of matching standard motion trajectory pairs between two robots to have a good guess for initial registration parameters. Figure 3.3 illustrates a typical registration between standard motions between two kinematic models. First, the initial scale factor s_i is defined as:

$$s_i = \frac{max(q) - min(q)}{max(p) - min(p)} \tag{3.16}$$

Since the order of the points in the trajectory is available, the initial parameters R_i and $\vec{t_i}$ can be computed by equations 3.13 and 3.10.

The iterative process defined above is described briefly as the algorithm 1.

After the registration process, we can transfer the motion for robot 1, M_1 to corresponding motion in robot 2, M_2 as:

$$M_2 = sRM_1 + \vec{t} \tag{3.17}$$

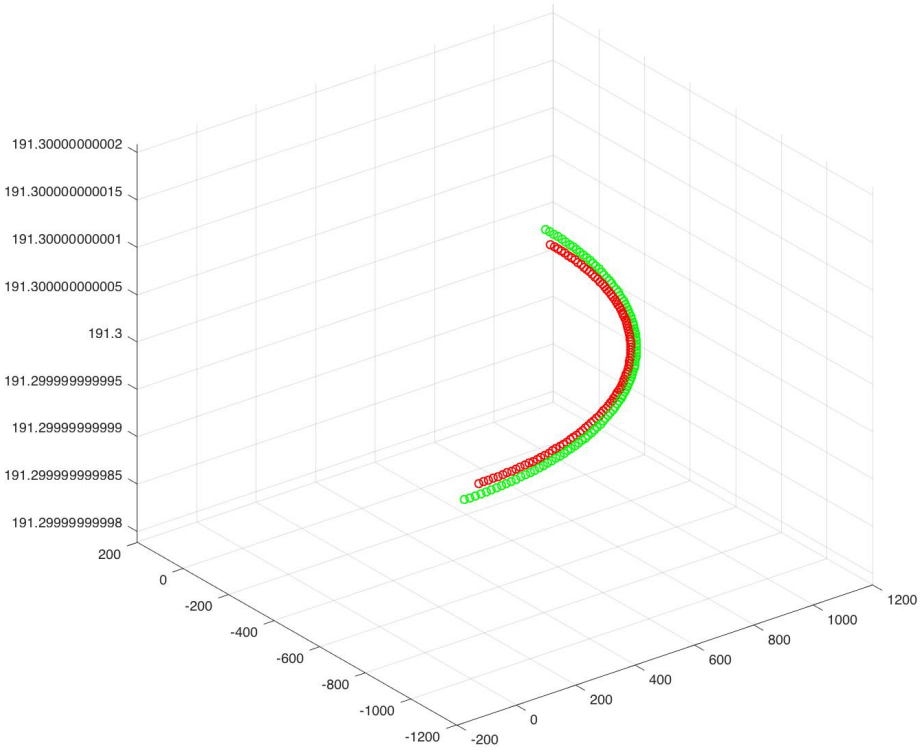

Figure 3.3: Standard Motions registration result between human and Baxtor. In this figure, the corresponding motion is straight right arm sweep.

Algorithm 1 Scale-ICP algorithm

Input: original data set P and Q, initial parameters $s_i, R_i, \vec{t}_i$ and terminating error ε

Output: optimal parameters $s, R, \vec{t}$

while $\sum_{\vec{p}_a \in P} \| sR\vec{p}_a + \vec{t} - \vec{q}_{c(a)} \|_2^2 < \varepsilon$ **do**

In iteration k, for $\vec{p}_a \in P_k$, find closest point set $Q_k \in Q$

Compute R_{k+1} by formula 3.13

Compute s_{k+1} by formula 3.14

Compute $\vec{t}_{k+1}$ by formula 3.10

end while

3.3 Dynamic Movement Primitives Learning

In this chapter, a motion dictionary is learned by implementing sparse approximation to the motion trajectories in the training set generated above. To recode complicated robotic behavior, compressive sensing algorithm tries to learn a dictionary from the original redundant information and use the combination of primitive elements in the dictionary to reconstruct the robotics skills. One simple, naive implementation of dictionary learning is PCA, which generates a complete set of bases. However, drawbacks of a PCA based method are noticeable. First, PCA is sensitive to the noise in the training set. In a addition, PCA which is linear does not fit the nolinear trajectory very well.

In order to realize the hidden pattern in the system, an over-complete set of bases is necessary to fit movement which is not contained in the training set. In addition, even if a relatively large dictionary is available, mixing most elements in the dictionary is not always helpful enough to find the significant factors (primary moves) for the data.

To have sufficient descriptive power yet remain efficient, the implemented dictionary learning algorithm should generate:

- An over-complete dictionary.
- Relatively short and clean code to recover the basic motion behavior information of robots.

Based on the requirements above, in our system, we found that the sparse coding method was adequate to compress the redundant movement information. We use the sparse auto-encoder from [18] to represent the original data:

In general, the error of sparse approximation can be defined as the sum of the average Euclidean distance from original variables to the linear combination approximate results:

$$J(a,b) = \frac{1}{m}\sum_{j=1}^{m} \|x^j - (\sum_{i=1}^{k} a_i^j \phi_i + b)\|^2 \tag{3.18}$$

In this equation, x represents m original data vectors. To compress these vectors, we use linear combination of k bases ϕ to fit x with the weighting vectors a and constant term b. One goal of our work was to get the minimum error from the linear combination of elements.

To avoid overfitting bases, regularization was introduced into the cost function. There are two possible regularization terms to use here:

- L_2 regularization term:

$$\frac{\lambda_1}{2}\sum_{i=1}^{n}\|a_i\|^2 \tag{3.19}$$

- L_1 regularization term:

$$\frac{\lambda_2}{n}\sum_{i=1}^{n}|a_i| \tag{3.20}$$

where λ_1 and λ_2 are coefficients. It can be argued which is better since the L_2 term provides a more smooth solution but L_1 term ensures its sparsity. In our work, L_2 term was added as regularization term in the cost function:

$$J(a,b) = \frac{1}{m}\sum_{j=1}^{m}\|x^j - (\sum_{i=1}^{k}a_i^j\phi_i + b)\|^2 + \frac{\lambda_1}{2}\sum_{i=1}^{n}\|a_i\|^2 \tag{3.21}$$

where λ is the regularization constant.

For the cost function $J(a,b)$, a neural network based autoencoder was implemented to find the optimal bases and corresponding weighting vectors. Figure 3.4 illustrates the structure of the encoder. Compared with a typical neural network, only three layers were used in our work: inputs, the hidden layer, and outputs. In the hidden layer, m neurons represent m elements in the resultant dictionary.

A neuron is the primary computational unit. For instance, for neuron n_1 the input z is a linear combination of elements from the input layer:

$$z = Wx + b \tag{3.22}$$

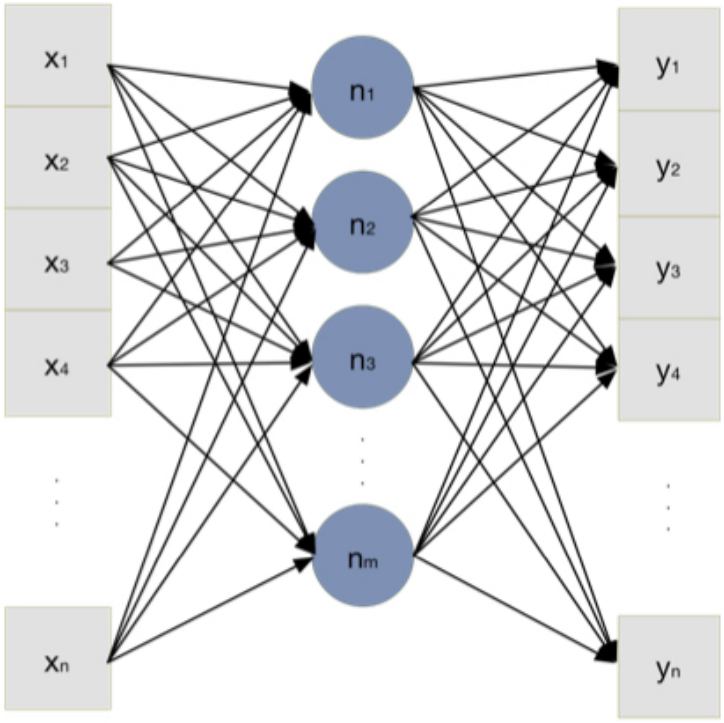

Figure 3.4: Structure of Neural Sparse Autoencoder

where W denotes the weights and b is the constant. To simulate the behavior of an actual neuron and to keep the active threshold around 0, a sigmoid function was applied to generate the output in this unit as shown in Figure 3.5.

$$S(z) = \frac{1}{1+e^{-z}} \tag{3.23}$$

Since a goal of our work is to represent the original data accurately, we set $x = y$ in the training process of this encoder network, which means that the input layer and output layer are identical.

Another requirement is sparsity in the movement primitives we generate. To achieve this, a sparse term $KL(\rho \| \widehat{\rho}_j)$ is designed to modify the original cost function.

$$KL(\rho \| \widehat{\rho}_j) = \rho log\frac{\rho}{\widehat{\rho}} + (1-\rho) log\frac{1-\rho}{1-\widehat{\rho}_j} \tag{3.24}$$

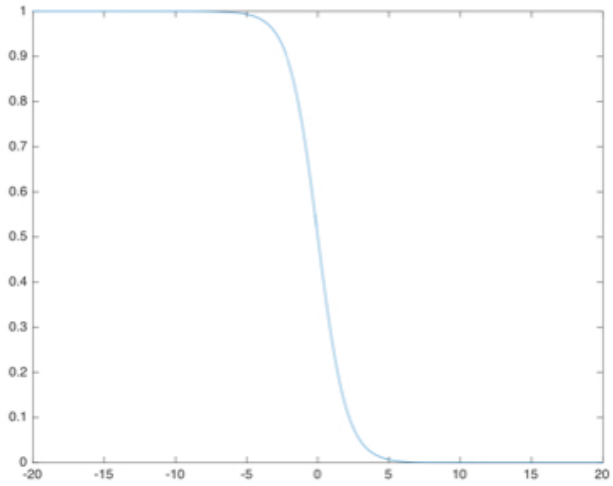

Figure 3.5: the sigmoid function

where ρ is the active threshold for the neuron.

Consequently, when this sparsity penalty is combined with L_2 term and the error defined above, the cost function of this neural network becomes:

$$J_{sparse}(a,b) = J(a,b) + \beta \sum_{j=1}^{m} KL(\rho \| \widehat{\rho}_j) \tag{3.25}$$

To find the minimum solution of the cost function $J_{sparse}(a,b)$, back propagation algorithm is applied:

Equation 3.22 and 3.23 define the feed-forward process in the Neural Network. To find the optimal weights for each neuron, back propagation [24] process is implemented to adjust the weights based on the difference between input layer and output layer.

The error of node j in output layer can be defined as:

$$E_j = S(z_j)(1 - S(z_j))(T_j - S(z_j)) \tag{3.26}$$

where T_j is the observed value for the node. In our work, T_j is the value of node j in input layer.

In the middle layer, error of node j is defined as weighted accumulation of nodes in next layer:

$$E_j = S(z_j)(1 - S(z_j)) \sum_k E_k W_{jk} \tag{3.27}$$

where W_{jk} represents the weight from current node j to node k in next layer. E_k is the error which is precomputed for node k.

To decrease the error described above, the weights for each node can be updated as:

$$\Delta W_{ij} = \lambda E_j S(z_i) \tag{3.28}$$

$$W_{ij} = W_{ij} + \Delta W_{ij} \tag{3.29}$$

corresponding constant term is:

$$\Delta b_j = \lambda E_j \tag{3.30}$$

$$b_j = b_j + \Delta b_j \tag{3.31}$$

where λ is value from 0 to 1 which represents learning speed. When λ is large, algorithm tends to have fast convergent rate, however the result may trap into a local minima.

When the maximum number of iterations is fixed or if we set a threshold for the system error, by applying back propagation iteratively, the optimal weighting vector can be generated from the structure so defined.

After the process of dictionary construction, a set of m bases ϕ is available. In the coding phase, the error is defined as:

$$\frac{1}{m} \sum_{j=1}^{m} \|x^j - (\sum_{i=1}^{k} a_i^j \phi_i + b)\|^2 \tag{3.32}$$

which can be solved by Lasso method [1].

Consequently, in this part, an optimal dictionary is created, and the original moves are recoded as the weighting vector.

3.4 DMP learning and Motion representation

In previous section, a dictionary of movement primitives was constructed using a sparse auto-encoder. Generated from the original trajectories, however , the movement bases in the skill dictionary are the record of the joints' 3D positions. To generalize the skill from point-point records to a general behavior, we apply the DMP system [8] to redefine the motion for each movement primitive we learn.

In the DMP system, the record of trajectory is treated as a linear spring system perturbed by an external forcing term:

$$\tau\dot{v} = K(g-x) - Dv + (g-x_0)f \tag{3.33}$$

$$\tau\dot{x} = v \tag{3.34}$$

Movement records are represented by these equations, where x is current position of joint and v is corresponding velocity; x_0 and g are the starting point and goal of the trajectory. K and D are constants that represent the spring coefficient and damping term of the system. f is the external force that drives the trajectory between the starting point and the goal.

$$f(t) = \frac{\sum_{i=1}^{N} \psi_i(t) w_i}{\sum_{i=1}^{N} \psi_i(t)} \tag{3.35}$$

Equation 3.4 acts as a fitting function for the external force, where ψ are the basis functions of the system, we choose a Gaussian model and w is an adjustable weight.

For the further development of our skill transfer dynamical system, a canonical equation

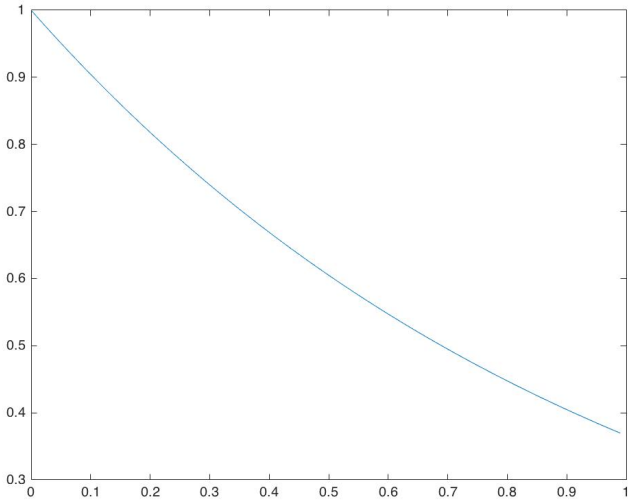

Figure 3.6: Canonical system in DMP algorithm

is introduced to make the system time-independent:

$$\tau \dot{x} = -\alpha_x x \tag{3.36}$$

where α_x is a constant which controls rate of convergence. Figure 3.6 shows an example canonical system. Starting from 1, the system reaches 0, which represents the goal of the trajectory, by Equation 3.36.

Therefore, the equation is reformed as:

$$f(x) = \frac{\sum_{i=1}^{N} \psi_i(x) w_i}{\sum_{i=1}^{N} \psi_i(x)} x(g - y_0) \tag{3.37}$$

Figure 3.7 shows bell-shaped curves we use for the nonlinear function approximator of the force term, which is defined as:

$$\psi_i(x) = e^{-\frac{(x-c_i)^2}{2\sigma_i^2}} \tag{3.38}$$

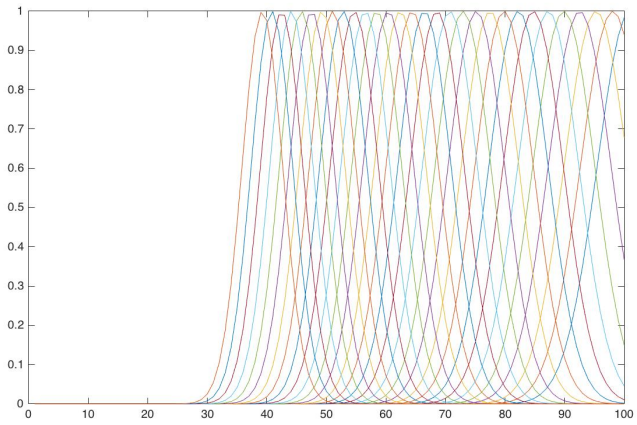

Figure 3.7: Kernel activation in DMP algorithm

Equation 3.33 can be reformed as:

$$f = \frac{-K(g-x) + D\dot{x} + \tau\ddot{x}}{g - x_0} \tag{3.39}$$

Locally Weighted Regression [1] is implemented to calculate the optimal parameters for the nonlinear function approximator to fit the force sequence defined in Equation 3.39.

3.5 Skill transfer and inverse kinematics

For a specific skill m_{input} in the source platform, similar to Equation 3.32, the error of recoding by the dictionary is defined as:

$$E_{fitting} = \|m_{input} - \sum_{j=1}^{N} W_j b_j\|^2 \tag{3.40}$$

where b denotes base in the dictionary. After finding optimal weighting vector W for m_{input}, the motion m_{input} is recoded as W, a low dimension vector, to transfer between robots.

The reconstructing motion in target platform can be executed as:

$$m_{result} = sR\sum_{i}^{N}(W_i DMP_i) + T \tag{3.41}$$

where s, R and T are registration parameters. DMP_i is the position sequence generated from the DMP system of base b_i.

To convert m_{result} for operation on the target platform, an inverse kinematics algorithm, such as the Jacobian inverse technique [25], can be implemented.

Chapter 4

SYSTEM DESCRIPTION

There are significant differences between industrial manipulators from different manufacturers and between different models from the same manufacturer. These include the number of joints, the DoF, the dimensions, the link configurations, the workspace, and the dynamics. Not the least, this includes the controllers, which are usually company proprietary, and the programming language, which is usually specific to one company's robots. A major goal of this work is to bridge those differences and to devise a systematic method for transferring skills (sequences of behaviors that comprise a motion or manipulation task) between different robots. To verify the skill transfer algorithm across heterogeneous platforms, our prototype system comprises three different models: a human body mechanical model from CMU motion database, the Yaskawa Motoman HP3JC, and the Rethink Robotics Baxter. Specific configurations are described in this chapter.

4.1 CMU Motion Database

The CMU Mocap lab contains 12 Vicon infrared MX-40 cameras. Humans wear a black jumpsuit and have 41 reflective markers taped on. The Vicon cameras see the markers in infra-red. Figure 4.1 demonstrates the view of marker set. The images that the various cameras pick up are triangulated to get 3D data. [26] To capture the mechanical features for input motion, only the 3D positions of joints are necessary. From a complete set of motion capture data we used only the right arm with 6 joints.

[1] http://mocap.cs.cmu.edu/info.php

Figure 4.1: Front and back view of marker set in CMU Motion Database[1]

4.2 Yaskawa Motoman HP3JC

The Yaskawa Motoman HP3JC is a compact vertical jointed-arm robot. Compared with other industrial robots, HP3JC is a compact robot that requires minimal installation space. Since the HP3JC is ideal for inspection/testing and research applications, we used this robot to test our skill transfer algorithm for a robot with 6 controlled axes. [27] It is programmed by moving it with a teach pendant or by transferring to it a program written in Yaskawa's proprietary language, INFORM. Figure 4.3 is a side view of the robot. The DH parameters for the HP3JC are listed in Table 4.1

[2]https://www.used-robots.com/motoman/used-hp3
[3]https://www.used-robots.com/images/robots/original/hp3jc-side.jpg

Figure 4.2: Yaskawa Motoman HP3JC[2]

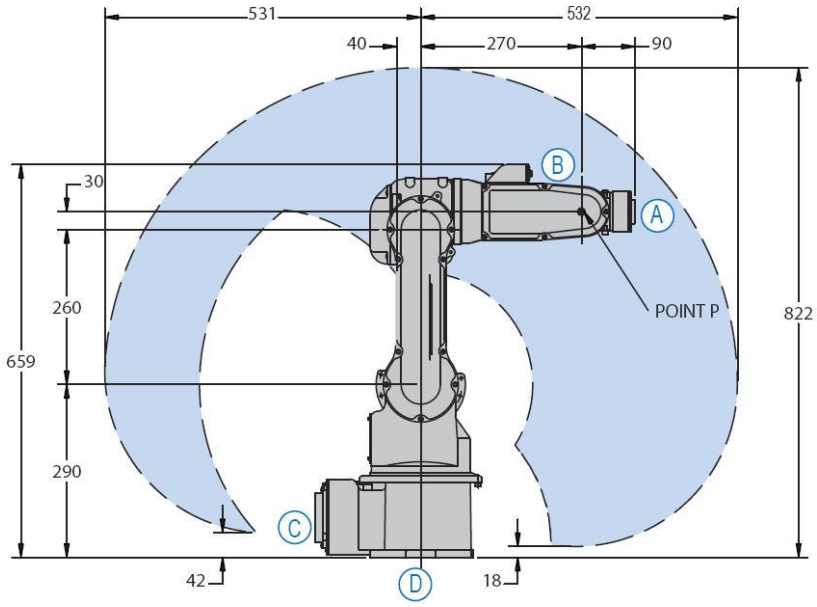

Figure 4.3: Side view of Yaskawa Motoman HP3JC[3]

Table 4.1: HP3JC joints constraints

Joint Number	θ	d	a	α
1	θ_1	157	0	$-\frac{\pi}{2}$
2	$\theta_2 - \frac{\pi}{2}$	0	260	$\frac{\pi}{2}$
3	θ_3	0	30	$-\frac{\pi}{2}$
4	θ_4	-270	0	$\frac{\pi}{2}$
5	θ_5	0	0	$-\frac{\pi}{2}$
6	θ_6	-135	0	0

4.3 Rethink Robotics Baxter

Baxtor is a relatively new robot for industrial automation. It's a "co-robot" designed to work in collaboration with people. With 7 DoF arms, Baxter is capable of executing a wide range of tasks. It can be programmed by moving its arms directly, or using the open-source ROS libraries with python, C, or C++. To test the transfer between different platforms, Baxter was treated both as a teacher robot and as a student respectively. Also, by implementing the algorithm between heterogeneous robots, the different mechanical structures, caused us to verify skill transfer under different geometric constraints. Figure 4.4 and figure 4.5 illustrate the physical structure of Baxter. Table 4.2 lists the DH parameters for its 7 joints.

Since it has 7 joints in each arm, Baxter is a redundant manipulator, more flexible than the HP3JC but more complex to program. This enabled us to test the transference of motion between 6 DoF and 7 DoF manipulators.

Table 4.2: Baxter joints constraints

Joint Name	θ	d	a	α
S0	θ_1	270.35	69	$-\frac{\pi}{2}$
S1	$\theta_2 + \frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
E0	θ_3	364.35	69	$-\frac{\pi}{2}$
E1	θ_4	0	0	$\frac{\pi}{2}$
W0	θ_5	374.29	10	$-\frac{\pi}{2}$
W1	θ_6	0	0	$\frac{\pi}{2}$
W2	θ_7	280	0	0

Figure 4.4: Rethink Robotics Baxter[4]

[4]http://www.rethinkrobotics.com/wp-content/uploads/2015/11/Baxter_Datasheet_Oct2015.pdf
[5]http://sdk.rethinkrobotics.com/wiki/Workspace_Guidelines

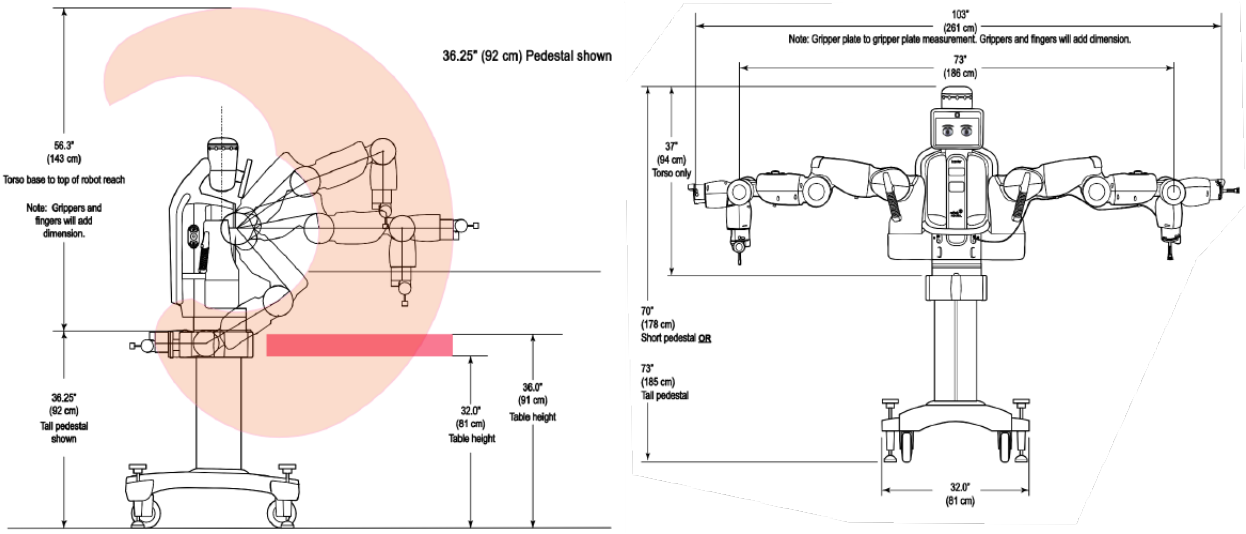

Figure 4.5: Front and side view of Baxter[5]

Chapter 5

EXPERIMENTS AND ANALYSIS

In last chapter, we described the database and the platforms we used in the experiments. Using the pipeline we described in Chapter 3, we performed experiments to verify our method. The movements we chose were extracted from the boxing and washing motion capture files from the CMU motion database as shown in Figure 5.1.

5.1 Potential Motion Trajectory Generation

In the motion trajectory part, a training set of 10,000 trajectories was generated by the randomized method in Chapter 3. Figure 5.2 shows the resultant trajectories. The generated trajectories covered many of the possible positions of the end joint. The figure shows that the robot workspace was covered to ensure the diversity and smoothness of generated trajectories.

Figure 5.1: Data from CMU motion database. Top: Boxing. Bottom: Washing window.

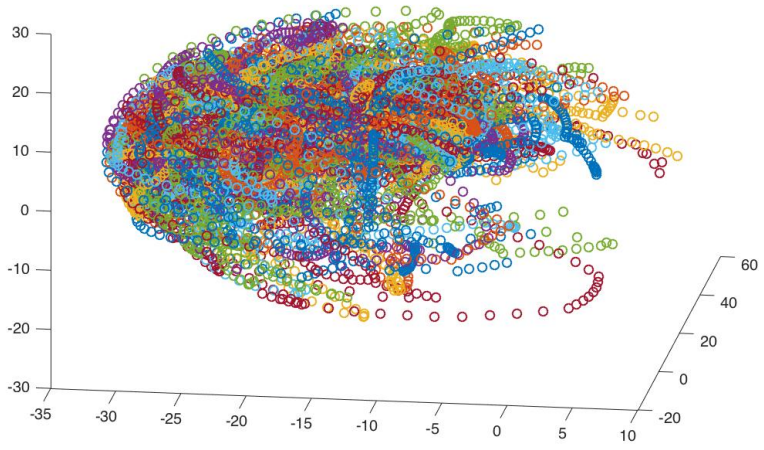

Figure 5.2: Motion data generated by our method

5.2 Motion Registration between Robots

To find the registration relation between the original model and the target robot platform, based on the motion analysis described in Chapter 4, the Scale-ICP algorithm was implemented. Figure 5.3 demonstrates point clouds from human skeleton model and Baxter before registration and corresponding registration result. The source of mis-matched points is the difference of sphere of activities between two models. Our ICP-based algorithm successfully find registration relation between platforms included scale factor, rotation and translation as follow:

$$R_{humanbaxter} = \begin{bmatrix} -0.9934 & -0.0813 & 0.0810 \\ -0.0813 & 0 & -0.9967 \\ 0.0810 & -0.9967 & -0.0066 \end{bmatrix}$$

$$T_{humanbaxter} = \begin{bmatrix} -74.4706 \\ 86.6437 \\ 490.6197 \end{bmatrix}$$

$$S_{humanbaxter} = 35.9$$

From motoman to baxter

$$R_{motomanbaxter} = \begin{bmatrix} -0.968 & 0.0159 & -0.249 \\ 0.0159 & -0.992 & -0.125 \\ -0.249 & -0.125 & 0.960 \end{bmatrix}$$

$$T_{motomanbaxter} = \begin{bmatrix} -187.83 \\ 387.44 \\ 403.77 \end{bmatrix}$$

$$S_{motomanbaxter} = 1.48$$

It's not necessary to find the registration relation between every pairs because of transitive relation between models.

5.3 Dynamic Movement Primitives Learning

For the learning of motion primitives, 25 bases were learned from the trajectories we generated. Figure 5.4 is the visualization of sparse coding result. In the left part, several tracks are randomly chosen from the training set. The right 25 images represent the bases we learned. Compared with training data, generated bases have typical patterns. Thus, by applying linear combination, the original data can be recovered.

Figure 5.5 demonstrates the resultant trajectory bases. Compared with original trajectories, the based learned is highly diverse and normalized. Figure 5.6 illustrates some details in the process of solving Lasso (least absolute shrinkage and selection operator) to generate

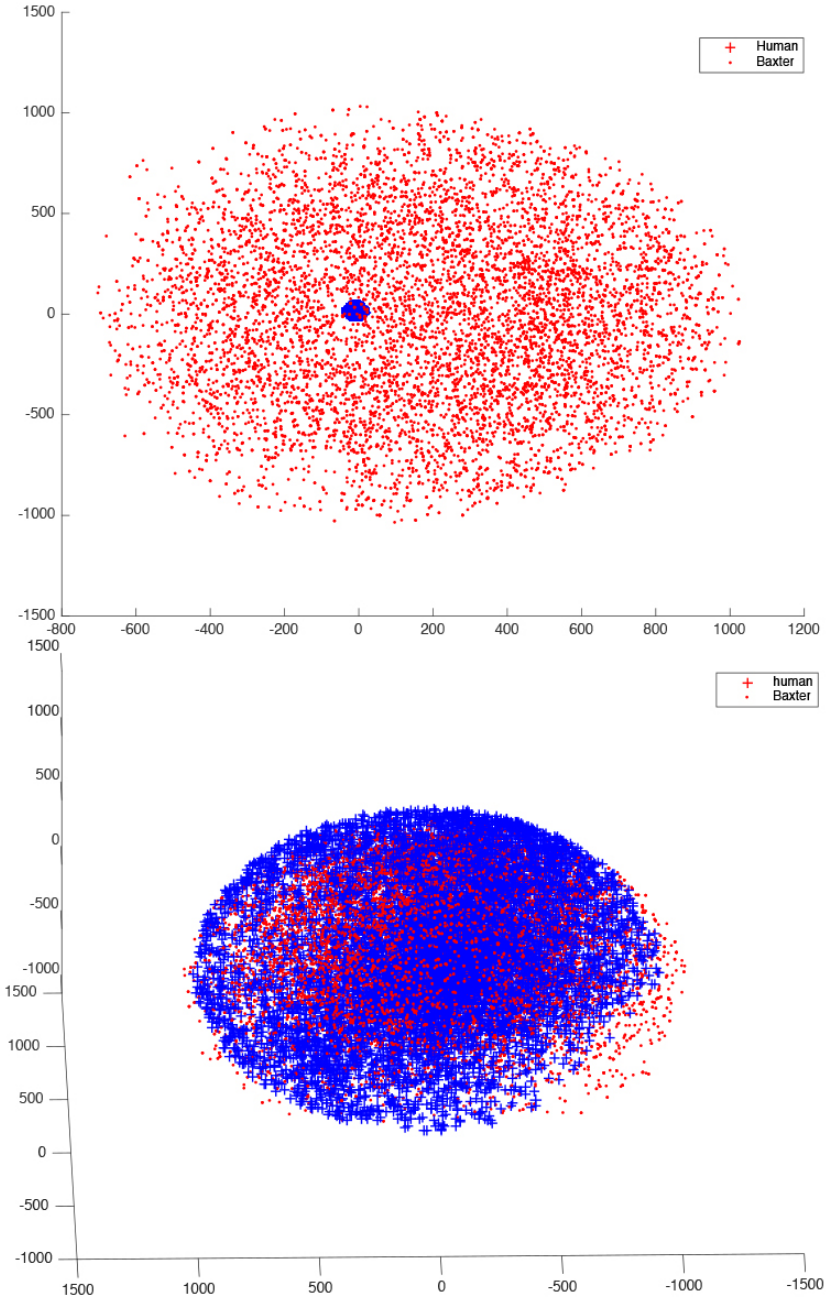

Figure 5.3: Pre-registration and resultant point cloud between human and Baxter

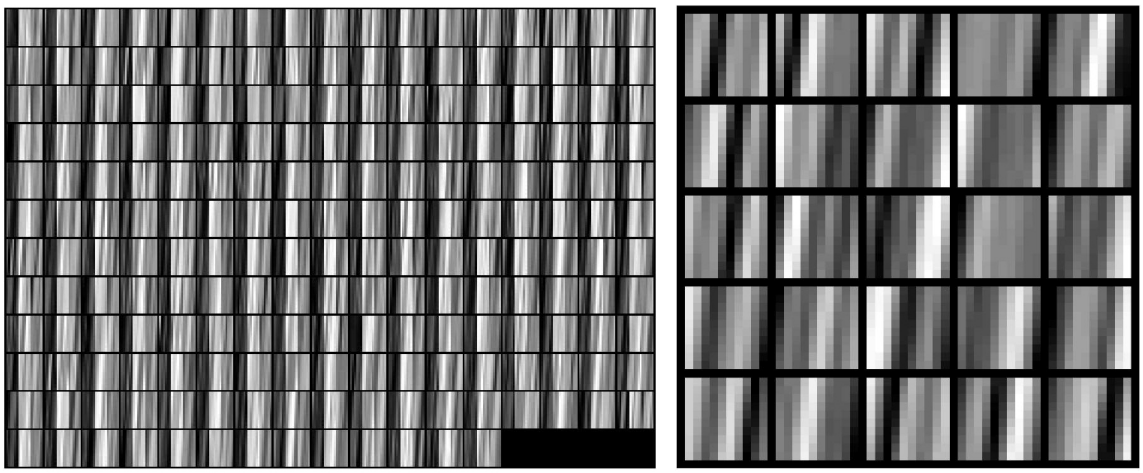

Figure 5.4: Visualization of sparse coding result. The original data is represented by the value of pixel.

the weighting vector: the name of the selected coefficient with a fitted value, the L_1 norm of a set of coefficients including the selected coefficient, and the index of the corresponding Lambda. The solving algorithm leads to find an answer which minimizes L_1 norm for each coefficient. Consequently, in the resultant weighting vector, only a few entries are non-zero to avoid overfitting.

In Figure 5.7, we compared original trajectory and fitting result from linear combination of movement bases. In Figure 5.5, it's easy to find that as prototype of movement, the bases are very smooth. Consequently, generated from linear combination of the bases, the fitting process acts as filter which eliminate noise in the original record.

In error estimation, we calculated mean squared error (MSE) for each trajectory and its fitting result. The fitting error distribution of 10k motion is demonstrated by Figure 5.8. As shown in Figure 5.8, sparse approximation is capable of representing and compressing original motion signals without a lot distortion.

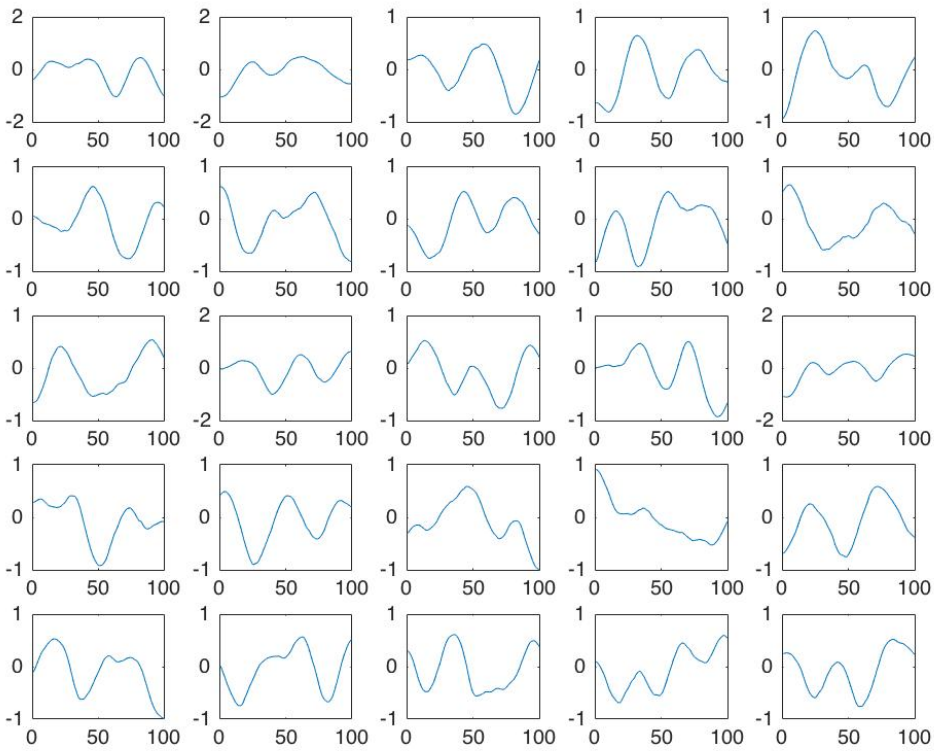

Figure 5.5: 25 learned bases of end-effector trajectories

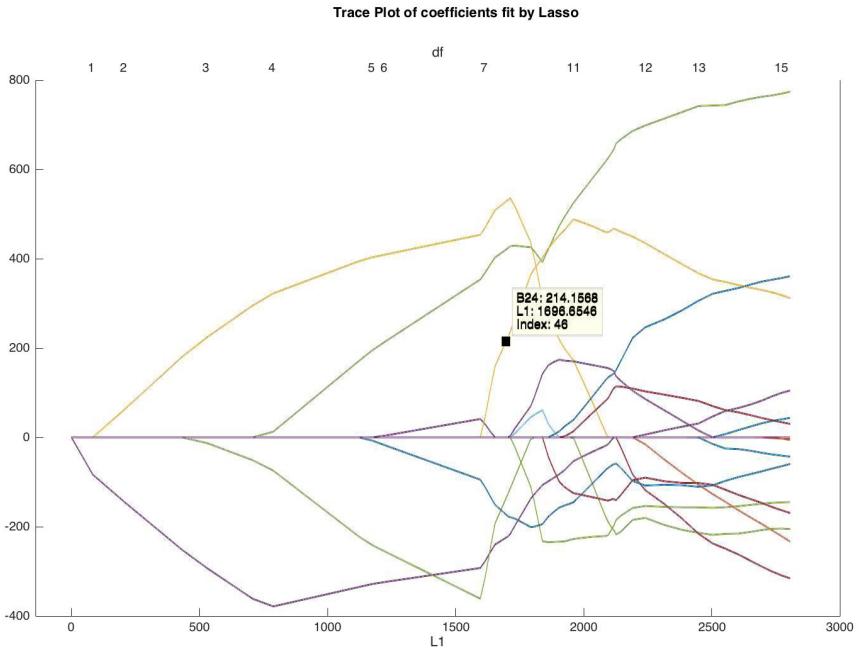

Figure 5.6: Process of solving Lasso [1]

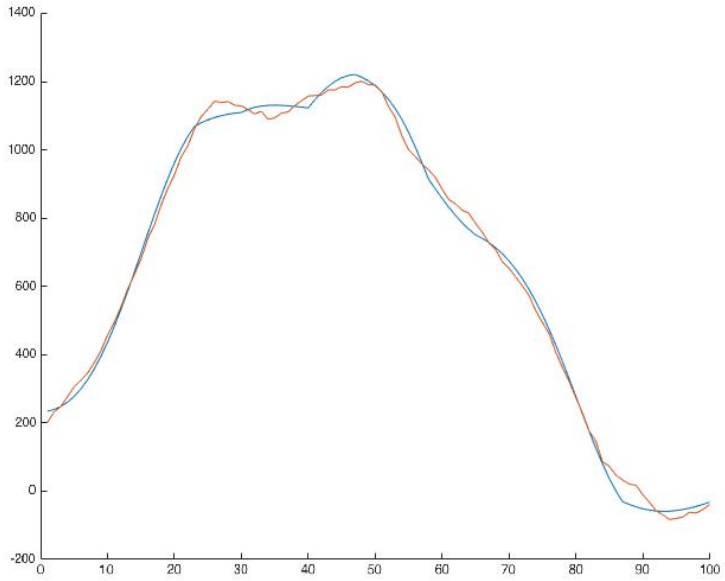

Figure 5.7: Original trajectory (red) and linear combination fitting result (blue) from bases

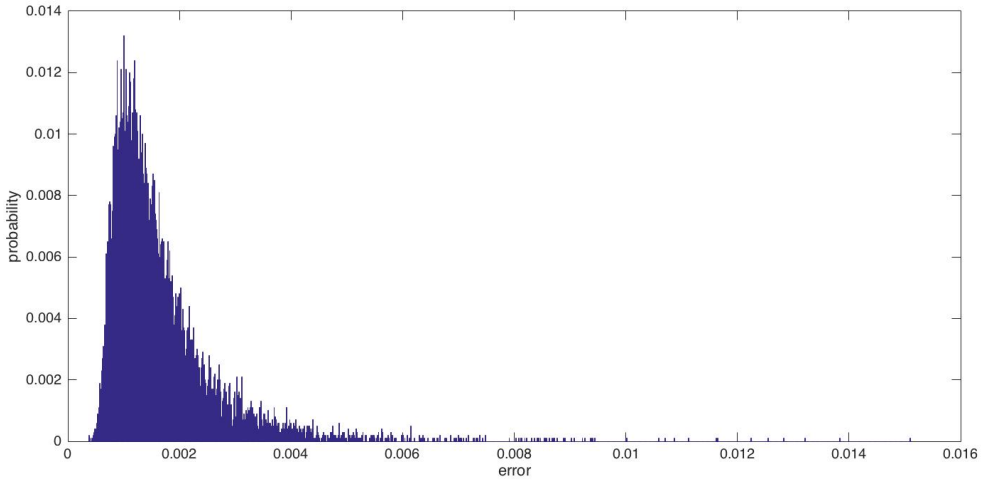

Figure 5.8: Error distribution of dictionary representation over 10,000 motion trajectories

5.4 DMP learning and Motion representation

Figure 5.9 demonstrates the how DMP system generalizes the movement in a dynamic system based on the bases we learned above. In the left figure, blue line is the raw record of the position. As the representation of the original data, red line from DMP fits the data well when they have same starting point and the goal. In the right figure, it shows that when we change the goal, DMP system is capable of setting attractor to the new goal and also keeping the behavior of inputed trajectory. Thus, when it's necessary to implement this skill to new destination, the DMP method enables we maintain the original behavior pattern.

5.5 Skill transfer and advantage in space complexity

Figure 5.10 is skill transfer result. The movement data is a snippet from motion of washing a window. Top left represents the original trajectory (red) and reconstruction (blue) in our system. By implementing registration, it's successfully transfered to Baxter (top right) and Motoman (bottom).

Another advantage our method is compression of redundant trajectory information.

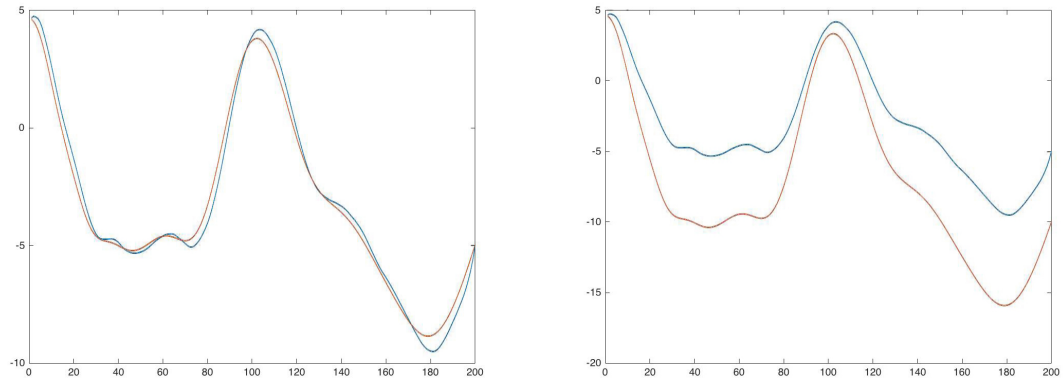

Figure 5.9: DMP of a movement base

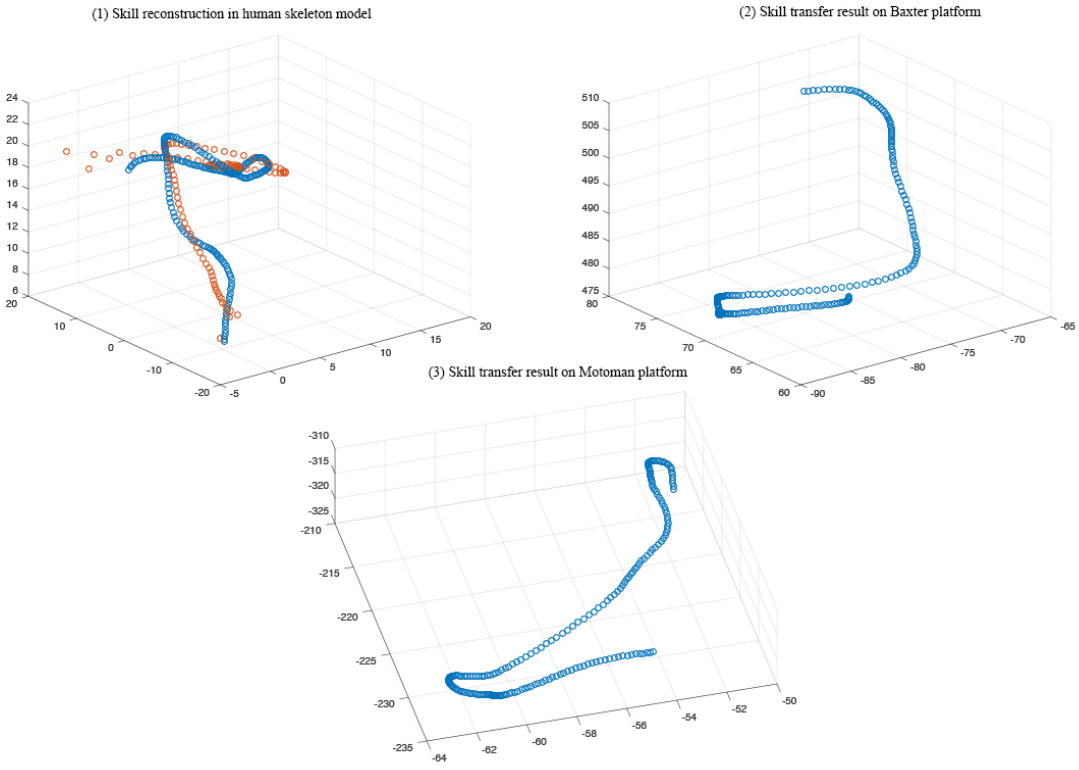

Figure 5.10: Skill transfer between human and Baxter, Baxter and Motoman

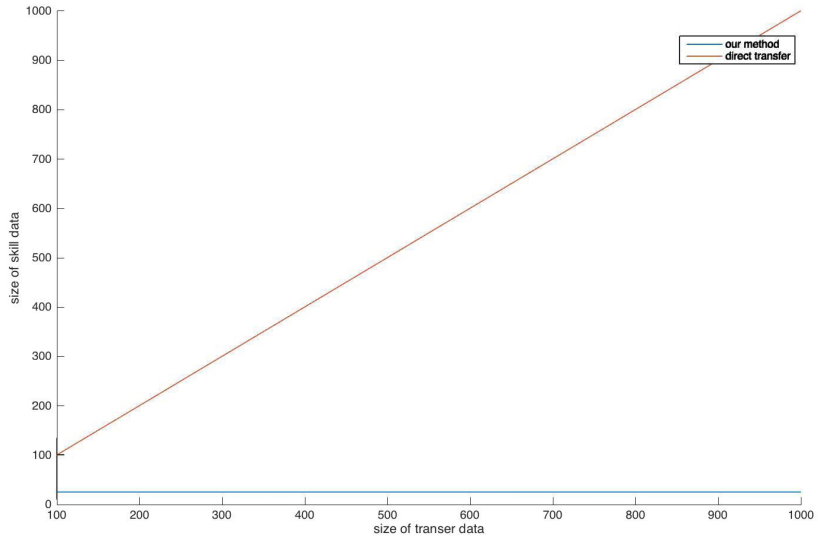

Figure 5.11: Difference of space complexity between direct transfer and our method

Figure 5.11 demonstrates the difference of space complexity between direct transfer and our method in the transfer process. If the number of bases is fixed, with increased size of a trajectory, our method keeps the space complexity constant. In contrast, if the trajectory data were to be transferred directly, the space complexity would increase linearly.

In Figure 5.12, we compared storage requirement between direct skill transfer and our dictionary-based transfer. In our method, the skill library consists of a fix-sized motion dictionary we learned above, and a set of weighting vectors. Compared with our method, direct transfer only keeps trajectories for all stored skills. In this figure, suppose we use n_1 bases to represent the original movement record. Each skill is stored in n_2 frames of position. In the skill library, if N skills were to be restored, the size of our system's requirement would be $n_1 \times n_2 + n_1 \times N$, and corresponding space requirement for direct transfer method would be $n_2 \times N$. Though the space complexity of direct transfer algorithm and our system are both linear $O(N)$. Consider the sparsity of the weighting vectors, $n_1 << n_2$. Consequently, the space requirement of our proposed system is much lower.

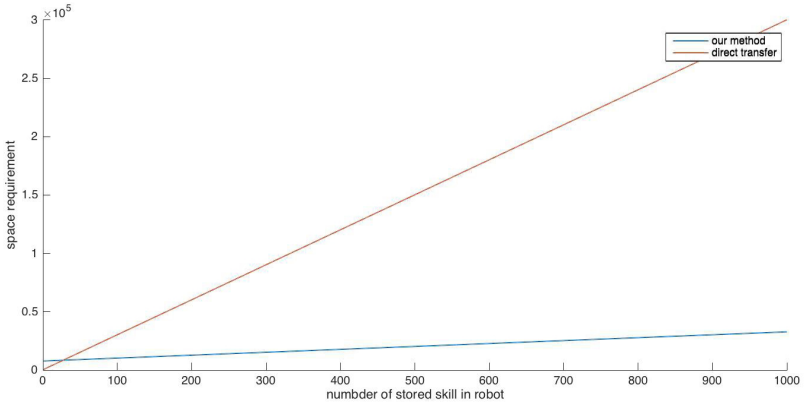

Figure 5.12: Difference of space requirement between direct transfer and our method

Chapter 6

CONCLUSION

In this work, we designed a system for skill transfer between industrial robots. The whole framework includes a Kinect based motion capture system and a skill representation and transfer algorithm. In the skill transfer part, the original skill is represented by a primitive motion dictionary. The experiments on the Rethink Robotics Baxter and the Yaskawa Motoman HP3JC with the CMU motion database show that the generated motion dictionary is capable of representing the original motion through a weighting vector. By implementing a transformation on the DMPs, the skill can be transferred between heterogeneous robots independent of their structural, mechanical, and dynamic differences. Moreover, compared with directly transferring trajectory data, our method can reconstruct high-dimensional movement from bases that have far lower dimensions than the input data.

We have achieved good transfer results between different models, however, there still exit many drawbacks in our system. First, in our system skills are represented by the trajectory of end-of-arm motion, since other joints' poses are not considered, there exits ambiguity in joints' angles. Furthermore, since error accumulates in the transfer process, registration error is potential risk in our system. Future work can be focused on error elimination by applying motion averaging technology. For instance, Govindu's work [28] exploits the information redundancy in a set of 3D scans by using the averaging of relative motions.

BIBLIOGRAPHY

[1] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[2] the International Federation of Robotics. History of industrial robots. https://docs.google.com/file/d/0B4JOD-8OZ5BEOUZQTUFRWTkwQXM/edit, 2012. [Online; accessed 08-Mar.-2017].

[3] Matt McFarland. Google officially reveals its latest robot. http://money.cnn.com/2017/02/28/technology/google-boston-dynamics-handle-robot/, 2017. [Online; accessed 08-Mar.-2017].

[4] robotiksistem.com. History of robotics. http://www.robotiksistem.com/robotics_history.html, 2009. [Online; accessed 08-Mar.-2017].

[5] Mengtang Li. *Skill Transfer between Industrial Robots by Learning from Demonstration*. PhD thesis, Vanderbilt University, 2016.

[6] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 763–768. IEEE, 2009.

[7] Freek Stulp and Stefan Schaal. Hierarchical reinforcement learning with movement primitives. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 231–238. IEEE, 2011.

[8] Elmar Rückert and Andrea d'Avella. Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems. *Frontiers in computational neuroscience*, 7:138, 2013.

[9] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[10] Tanveer Abbas and Bruce A MacDonald. Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3816–3821. IEEE, 2011.

[11] Mark A Davenport, Marco F Duarte, Yonina C Eldar, and Gitta Kutyniok. Introduction to compressed sensing. *Preprint*, 93(1):2, 2011.

[12] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

[13] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[14] http://research.cs.wisc.edu. Asf/amc file description. http://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/ASF-AMC.html, 2017. [Online; accessed 08-Mar.-2017].

[15] MICHAEL J. MILLER. Primesense: Motion control beyond the kinect. http://forwardthinking.pcmag.com/gadgets/282321-primesense-motion-control-beyond-the-kinect, May 2, 2011. [Online; accessed 08-Mar.-2017].

[16] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[17] Shihui Ying, Jigen Peng, Shaoyi Du, and Hong Qiao. A scale stretch method based on icp for 3d data registration. *IEEE Transactions on Automation Science and Engineering*, 6(3):559–565, 2009.

[18] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

[19] Richard S Hartenberg and Jacques Denavit. A kinematic notation for lower pair mechanisms based on matrices. *Journal of applied mechanics*, 77(2):215–221, 1955.

[20] Saeed B Niku. *Introduction to robotics: analysis, systems, applications*, volume 7. Prentice Hall New Jersey, 2001.

[21] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.

[22] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[23] J Michael Fitzpatrick. Fiducial registration error and target registration error are uncorrelated. In *SPIE Medical Imaging*, pages 726102–726102. International Society for Optics and Photonics, 2009.

[24] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[25] wikipedia.org. Inverse kinematics. https://en.wikipedia.org/wiki/Inverse_kinematics, 2017. [Online; accessed 08-Mar.-2017].

[26] mocap.cs.cmu.edu. Cmu motion database description. http://mocap.cs.cmu.edu/info.php, 2017. [Online; accessed 08-Mar.-2017].

[27] www.motoman.com. Yaskawa motoman hp3jc specification. https://www.motoman.com/hubfs/HP3JC-1.pdf?t=1489503085836, 2017. [Online; accessed 08-Mar.-2017].

[28] Venu Madhav Govindu and A Pooja. On averaging multiview relations for 3d scan registration. *IEEE Transactions on Image Processing*, 23(3):1289–1302, 2014.