공학박사학위논문

가우시안 프로세스 동적 모델을 이용한 동작 인식과 계획

Motion Recognition and Planning Using Gaussian Process Dynamical Models

2017 년 8 월

서울대학교 대학원

기계항공공학부

안 병 철

Motion Recognition and Planning Using Gaussian Process Dynamical Models

Byungchul An

Submitted in Partial Fulfillment of The
Requirements for The Degree of

Doctor of Philosophy

in the
School of Mechanical and Aerospace Engineering

at
Seoul National University

August 2017

ABSTRACT

Motion Recognition and Planning Using Gaussian Process Dynamical Models

by

Byungchul An

School of Mechanical and Aerospace Engineering
Seoul National University

In this thesis, we deal with the problems that the robot copes with unstructured environments. Examples of such environments are obstacles that robots should avoid and terrain features that are closely related to the intentions of the wearer of an exoskeleton robot. We make robots to avoid obstacles through path planning algorithms in joint space and its low-dimensional space. We also estimate human motion intentions caused by terrain features using machine learning techniques.

First, we propose an algorithm based on Gaussian process dynamical models (GPDM) to estimate motion intention of the wearer of exoskeleton robot. For the

observed short time series input values, the corresponding low dimensional space coordinates are obtained via Gaussian process regression. The similarity for each model is expressed in the form of the logarithm of the conditional probability distribution of observed values and its low-dimensional coordinates given the training data. This similarity is compared to estimate the most likely motion. We validate our algorithm through physical experiments using an exoskeleton robot prototype and motion tracking system.

Next, we propose a rapidly-exploring random tree (RRT) algorithm that adaptively determines an appropriate stepsize. Using a standard operator norm inequality and the forward kinematics equations expressed as the product of exponentials, we derive an approximate bound on the Cartesian displacement of the open chain tip for a given joint space displacement. Using this inequality bound, we adaptively determine the stepsize for a given minimum obstacle size. We verify our algorithm by numerical experiments using a ten-dof planar open chain robot and a seven-axis industrial manipulator.

Finally, we propose a path planning method in a low-dimensional space that generates a human-like motion by learning the human demonstration motion using GPDM. We extend the above inequality to the inequality between displacement in the low-dimensional space and displacement of each links in the workspace. We use this to map the obstacles defined in the workspace to the low-dimensional space based on the uniform sampling. In addition, we define a measure based on the GPDM kernel function to measure the similarity between the learned motion and the newly generated motion. We validate the proposed method by applying it to a simulator and an actual robot.

Keywords: Gaussian process dynamical models, path planning, path planning in latent space, motion intention recognition, machine learning, rapidly-exploring random tree, adaptive stepsize, operator norm.

Student Number: 2009-20694

Contents

List of Tables

List of Figures

1

Introduction

The most basic thing to be guaranteed when robot moves is that it does not collide with the obstacle, the environment. Because robots generally move at high speed and are heavy, if robot collides with the surrounding environment, it can damage not only property but also injure people. Therefore, obstacle avoidance ability is the most basic condition that robots should have. On the other hand, in the case of an exoskeleton robot that directly interacts with humans and assists the work performed by humans, contact with the surrounding environment is inevitable. In this case, since the robot moves in response to human will, how to find a human intention to respond to the environment is an important issue rather than avoiding obstacles. In particular, the lower-limb exoskeleton is in contact with the terrain at every step, so human intention and terrain are closely related. This thesis address the problems of the functions that robots should have as a first priority: obstacle avoidance and recognizing human motion intention.

First, we address the problem of real-time estimation of the motion intention of the lower-limb exoskeleton wearer. Lower-limb exoskeletons are wearable devices

that deliver additional power to the wearer's hips, knees, or ankles and are being developed for rehabilitation, daily activities assistance, or military purposes. In everyday life, we perform not only walking on the flat ground but also walking up/down the stairs, walking up/down ramps, sitting and standing. Exoskeletons should be able to distinguish between these actions because the appropriate control law for each movement type will be different. On the other hand, due to the kinematic structure and the various sensors, the data will naturally be of higher dimensionality. Our method utilizes Gaussian process dynamical models (GPDM) [1, 2] to efficiently represent common features from data belonging to the same movement type into a low-dimensional latent space trajectory and to capture the dynamic characteristics with the second order Markov process. The similarity between new consecutive three frames observation data and each movement type is expressed in the form of Gaussian conditional probability distribution and estimates the most similar movement type therefrom. We validated our algorithm through physical experiments with a prototype.

Most planning algorithms, including rapidly-exploring random tree-based (RRT-based) algorithms that we will discuss in this thesis, perform path planning in a joint configuration space. Since RRT is a sampling-based algorithm, it generates paths by connecting randomly generated nodes in a joint space. The distance between adjacent nodes is determined by the user-specified stepsize, and the algorithm may not detect the collision between two nodes according to the size of the stepsize. The collision check is performed in the Cartesian space occupied by the robot link. Therefore, stepsize should be chosen with appropriate value considering the size of workspace object. To do so, using the forward kinematics expressed as the product of exponentials and a standard operator norm inequality, we derive an approximate bound on the Cartesian displacement of the open chain tip for a given

joint space displacement. From this inequality, the appropriate stepsize is determined adaptively in terms of the size of the smallest workspace object and based on this, an adaptive stepsize RRT algorithm is proposed. Our algorithm eliminates the need to manually determine the stepsize through time-consuming trial and error process. Our adaptive stepsize algorithm has been validated through extensive numerical experiments with a ten-dof planar open chain and a seven-axis industrial arm manipulator.

In recent years, robots are not limited to the industrial and manufacturing sectors but are increasingly developing in the direction of human daily life. These robots are required not only to avoid obstacles but also to perform human-like motions in order to improve familiarity with people. However, since these movements are very complex and the robots are high dimensional, such movements are difficult to analyze or express mathematically. To deal with this problem, recent researchers are trying to solve these problems through statistical machine learning techniques. In statistical machine learning methods, the object of learning is human motion. Because these motion data are high dimensional, the dimensionality reduction is employed before or simultaneously with learning. We have attempted to address this problem using GPDM. By GPDM learning, the dimensionality of human demonstration motion data was reduced; regions, where collisions occur in this reduced lower dimensional space, are mapped out by the inequality bound we derived from the previous study; we defined a similarity measure between the learned and newly generated paths. Based on these, we proposed a motion planning method that avoids obstacles while performing a similar motion to the human in low dimensional space.

We now describe in more detail main contributions of this thesis.

1.1 Contributions of This Thesis

1.1.1 GPDM-Based Human Motion Intention Recognition for Lower-Limb Exoskeleton

In order for the lower-limb exoskeleton to help people efficiently motion intention of a person (e.g., walking, walking up/down the stairs, squatting, etc.) should be correctly estimated. Because the control law for walking on the flat ground will be different from the control law for going up the stairs. Studies predicting human motion intention have been mainly based on electromyography (EMG) or electroencephalogram (EEG) [3], [4], [5]. However, attaching these sensors to human bodies is very cumbersome and sensitive to a variety of environmental factors, making them unsuitable for practical purposes, especially for military use.

However, since EMG or EEG signals are measured before the muscles of the human body activate, basic machine learning algorithms such as linear discriminant analysis (LDA) [6] and support vector machine (SVM) [7] have well estimated human motion intentions. For this reason, it is very challenging to estimate human motion intent without EMG or EEG signals. Actually, in our preliminary studies, SVMs without EMG or EEG signals did not estimate human motion intent at all. While human motion data reside in the continuous time domain, LDA or SVM may not reflect the dynamic nature of this data. We used GPDM to learn the dynamic characteristics of such data.

Generally, the lower-limb exoskeleton is composed of a rigid body in order to assist the movement without disturbing the movement of the person as much as possible, and therefore has many joints. In order to estimate the current state of the robot and apply appropriate control law, there are various sensors such as AHRS, IMU, pressure sensors as well as encoders. There are also various levels

of noise. We show that GPDM can extract meaningful features from the high dimensional data composed of various sensor signals, which are corrupted by various sensor noises.

Our motion recognition method learns K GPDMs given each motion data for K total motion intentions. Given new observation data, the latent space coordinate corresponding to this observed data is found using Gaussian process regression. Then calculate the log conditional density for each model using both observation data and the corresponding latent space coordinates. Among the K log conditional densities, the model having the maximum value is estimated as the intention of the motion.

In order to show the effectiveness of our algorithm, we compared ours with a HMM-based algorithm, which is one of the algorithms that can reflect the dynamic characteristics of the data. We performed two case studies. The first is the data obtained by capturing the movement of each part of the human body using the VICON motion capture system and then converting it into the joint angle value. The second is the data of the subjects wearing the lower-limb exoskeleton with sensors except for the actuators. Movement types include walking on flat ground, walking up/down the stairs, and walking up/down an incline. Those types are the motions we perform in our daily lives [8]. For both data, the GPDM-based method showed better or similar level of accuracy of motion intention recognition than the HMM-based method.

1.1.2 An Adaptive Stepsize RRT Planning Algorithm for Open Chain Robots

Planning algorithms that rely upon the randomly-exploring random tree (RRT) [9] are some of the most widely used probabilistic sampling-based methods for robot

path planning today. The basic RRT algorithm is efficient and robust, easy to implement, and lends itself well to generalization to more complex robots (e.g., closed chains) and to diverse robot and task constraints (e.g., velocity and actuator limits, dynamic balance requirements). Many extensions and improvements to the basic RRT algorithm have been proposed in the literature; see, e.g., [10], [11], [12] and the references cited therein.

A typical RRT-based algorithm requires that the user specify the stepsize parameter. As implied by the name, the stepsize determines the distance between the current node and a newly generated node: a larger stepsize means that the spacing between a pair of connected nodes is larger, implying that a feasible path can generally be found with a sparser tree (and thus more quickly). If the stepsize is set too large, however, the resulting path may not be feasible: because collision checking is usually performed only at the nodes for the reason of that collision checking is the most time-consuming process in sampling-based planning, possible collisions with small objects (i.e., those that are small enough to fit between two connected nodes) may go undetected. The effects of the choice of stepsize on RRT performance are further investigated in [12]. If the size of the smallest workspace object is known in advance, then choosing the largest possible stepsize that still prevents these smallest objects from fitting between two connected nodes is one practical means to ensure collision-free paths. Choosing an appropriate stepsize is often a time-consuming trial-and-error process that makes RRT, despite its simplicity and well-documented advantages, challenging to use in practice.

Instead of choosing appropriate stepsize, varying stepsize has been studied in several literatures. The RRT-Connect [13] continues to extend for a fixed stepsize in the direction of extending the new node until a collision occurs or the goal is reached. The algorithm was presented based on the heuristics that extending the

tree in this way significantly improves the speed of RRT algorithm. The RRT-Connect does not actually change the stepsize, but the effect is the same as changing the stepsize. Further heuristics about how far to grow a tree is reported in [14]. They reported that it is better to keep about 90% of the valid part when extending the tree in the same way as the RRT-Connect. Because the valid part close to the collision can be difficult to expand further. For a rigid body such as a UAV and mobile robot, methods of adaptively varying stepsize have been studied [15], [16]. In [15], a randomly sampled node and the nearest node are connected directly if there is no obstacle between these nodes. On the other hand, an algorithm is proposed to obtain more information about a given size map during the same amount of time[16]. In that algorithm, stepsize increases with the number of nodes from the root node to the current node.

In this study we propose an adaptive stepsize RRT planning algorithm for open kinematic chains. Like most RRT planners, ours constructs a tree in the joint configuration space, while collision checking is done in the Cartesian space occupied by the robot links. Using a standard inequality on the matrix norm induced from a vector norm, we derive an approximate bound on the Cartesian displacement of the open chain tip for a given joint space displacement. This bound, which can be computed in real-time and is useful for a range of motion planning contexts, is then used to compute, for a given robot configuration, approximate bounds on the maximal deviations of the points on each of the robot's links (each link is modeled as a convex polytope). In this regard the product of exponentials forward kinematics formula [17] plays an essential role in the derivation and efficient computation of these bounds. Not having to manually set the stepsize parameter through a time-consuming trial-and-error process is an important feature of our algorithm.

1.1.3 A Gaussian Process Dynamical Model-Based Planning Method

As robot technology becomes more and more closely related to real life, robots are required to behave like human beings. Since learning from demonstration (LfD) simply allows the end user to program the robot by showing how the human beings perform the given task, movements which are very difficult to express mathematically such as human-like motions can be transmitted to the robot relatively easily.

Conventional LfD methods are HMM-based methods [18], [19], [20], [21], [22]. The main problem with HMM-based methods is the existence of unnatural discontinuities in describing the data and insufficient the number of data compared to the complexity of the model.

In general, low-dimensional space is used to solve the problems caused by the high dimensionality of robot motion. For many reasons including safety, obstacles should be considered when generating robot motion, but it would be more effective if obstacles could be efficiently brought into the low dimensional space. However, as far as we know, no research has been done to bring the obstacles in the workspace to the latent space.

In this study, using latent space constructed by GPDM, we proposed a planning algorithm that can generate motion similar to learned motion while avoiding obstacles in latent space. With GPDM, it is possible to continuously bring learning motion into low dimensional space with only a small amount of data. To bring obstacles in workspace into latent space, we use the Cartesian displacement bound proposed in Chapter 4. We applied the GPDM mapping function to the extended version of the inequality defined between joint space displacement and Cartesian space displacement to latent space displacement and Cartesian space displacement

relation. In order to exploit the learned motion, a similarity measure between the learned motion and the generated motion is defined in terms of the GPDM kernel function. This similarity measure is a kind of potential field that is differentiable. Finally, the VF-RRT is used to generate a motion by utilizing the vector field obtained by differentiating this potential field while avoiding obstacles.

We verified our algorithm by simulating the motion of grasping a cylinder-shaped object on a table and applying it to an actual robot.

1.2 Organization

In Chapter 2 we first present Gaussian processes and their extensions. We first examine Gaussian process regression. Then we examine Gaussian process latent variable model and it is followed by Gaussian process dynamical model. We then briefly review the forward kinematics of serial open chain relying on the product of exponentials.

In Chapter 3 we propose a motion intention recognition algorithm for lower-limb exoskeleton. We encode given training data belongs to the same movement type through a GPDM. Then the similarity of a newly observed data to the given training data is calculated as a probabilistic model. We present two case studies with physical experiments data obtained from a prototype lower-limb exoskeleton and motion tracking system.

In Chapter 4 we propose an adaptive stepsize RRT planning algorithm for open chain robots. First, we derive a Cartesian displacement bound for open chains. Then we provide our detailed algorithms for determining an adaptive stepsize using the bound and an idea to speed up our algorithm derived from the bound. Finally, we present our numerical experimental results involving a ten-dof planar

open chain and a seven-axis industrial robot arm.

In Chapter 5 we provide a planning algorithm that exploits GPDM latent space. We extend the bound derived in Chapter 4 in order to determine the sampling resolution to map out the regions where collision occurs. A similarity measure is defined between learned paths and newly generated path. After numerical studies, we apply it to actual robot.

In Chapter 6 we conclude this thesis with a summary of our main results, and discuss further directions for future work.

2

Preliminaries

2.1 Introduction

In this chapter, we first review Gaussian process and its extensions. As extensions of the Gaussian process, widely used machine learning techniques are Gaussian process regression, Gaussian process latent variable model, and Gaussian process dynamical model. In this paper, Gaussian process dynamical model, which is one of the extensions of a Gaussian process, is used to efficiently compress and express high dimensional robot motion data. We then briefly review the forward kinematics of open chains expressed in the form of the product of exponentials. This forward kinematics equation is used to derive the main idea of Chapter 3.

2.2 Gaussian Process

2.2.1 Gaussian Process Regression

Gaussian process (GP) is a statistical distribution and can be defined as follows. For a given set of data $\{X_t\}_1^N$, if all subsets except the empty set of this set satisfy the multivariate Gaussian distribution, this set is called a Gaussian process. The Gaussian process is a probability distribution over the function, different from the Gaussian distribution, which is a probability distribution over the vector space, and is expressed as follows:

$$f(x) \sim GP\left(m(x), k(x, x')\right), \tag{2.2.1}$$

where m is a mean function, k is a covariance function, and characteristics of GP are determined by the covariance function.

Gaussian process regression (GPR)[23] models a given input/output pair $\{\mathbf{x}_i, y_i\}_1^N$ as a GP as follows,

$$y \sim GP\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right), \tag{2.2.2}$$

unknown function $y = f(\mathbf{x})$ is obtained as in the form of Gaussian distribution. For a new input $\mathbf{x}^*$, new output y^* can be obtained as a conditional probability distribution for existing data.

Consider the following function:

$$y = f(\mathbf{x}) + \eta \tag{2.2.3}$$

where $f : \mathbb{R}^D \to \mathbb{R}$ is a nonlinear function, $\mathbf{x} \in \mathbb{R}^D$, and η is a zero-mean isotropic Gaussian white noise process. Function f can be expressed as a linear combination of scalar basis functions $\phi_i(\mathbf{x}) \in \mathbb{R}$:

$$f(\mathbf{x}) = \sum_i^m a_i \phi_i(\mathbf{x}) \tag{2.2.4}$$

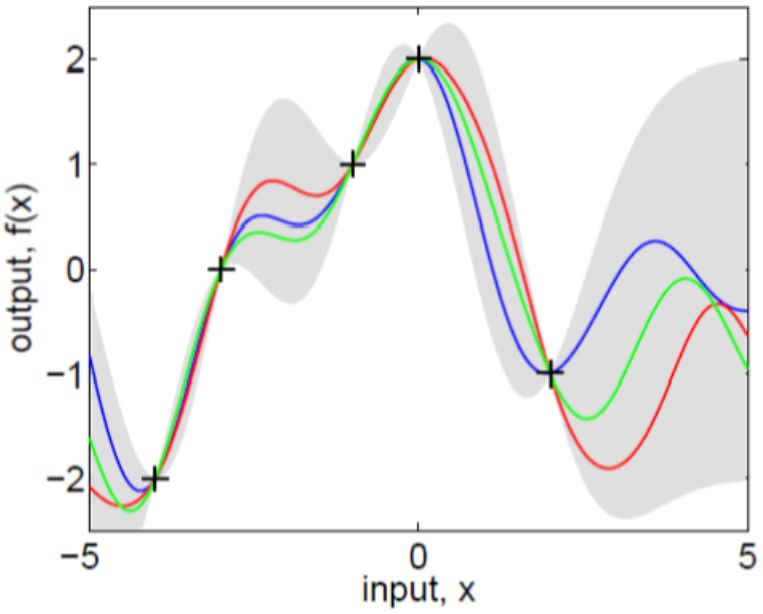


Figure 2.1: Gaussian process regression. [1]

where $A = [a_1, a_2, \cdots, a_m]^T \in \mathbb{R}^m$ is a weight vector of the basis function. Assuming Gaussian prior to A, i.e. $p(A) = \mathcal{N}(0, I_m)$, I_m is the $\mathbb{R}^{m\times m}$ identity matrix, one can get a probability distribution for the function $f(\mathbf{x})$. For arbitrary input data $\mathbf{x}$, mean and variance of $f(\mathbf{x})$ are expressed as

$$E[f(\mathbf{x})] = \phi^T(\mathbf{x})E[A] = 0 \tag{2.2.5}$$

$$E[f(\mathbf{x})f(\mathbf{x}')] = \phi^T(\mathbf{x})E[AA^T]\phi(\mathbf{x}') = \phi^T(\mathbf{x})\phi(\mathbf{x}'), \tag{2.2.6}$$

since $\phi^T(\mathbf{x})\phi(\mathbf{x}')$ is a symmetric positive semi-definite function, it can be expressed as any kernel function $k(\mathbf{x}, \mathbf{x}')$.

For a given set of data $\{\mathbf{x}_i, y_i\}_1^N, X = x_{i1}^N, Y = y_{i1}^N$, marginal distribution on

[1] C. E. Rasmussen, *Gaussian processes for machine learning,* 2006.

Y is given as follows:

$$p(Y|X,\theta) = \mathcal{N}(0,K) \tag{2.2.7}$$

where $K \in \mathbb{R}^{N\times N}$ is a kernel matrix whose elements $K_{i,j}$ are defined by the following kernel function $k(\mathbf{x}_i,\mathbf{x}_j)$ and θ is the kernel hyperparameter. The most widely used kernel in the Gaussian process is the radial basis function (RBF) kernel, which is the form of squared exponential function

$$k(\mathbf{x}_i,\mathbf{x}_j) = \theta_1 \exp\left(-\frac{\theta_2}{2}||\mathbf{x}_i-\mathbf{x}_j||^2\right) + \theta_3\delta_{i,j}. \tag{2.2.8}$$

In the above equation, $\theta = [\theta_1,\theta_2,\theta_3]$ is the kernel hyperparameter and $\delta_{i,j}$ is the Kronecker delta function.

Learning

Learning Gaussian process regression is to estimate kernel hyperparameter θ from the given input/output data pair $\{X,Y\}$. Estimation of the kernel hyperparameter is done by optimizing log-likelihood function $\log p(Y|X,\theta)$ with respect to θ. Log-likelihood function $\log p(Y|X,\theta)$ is given as

$$\log p(Y|X,\theta) = -\frac{1}{2}Y^T K^{-1}Y - \frac{1}{2}\log|K| - \frac{N}{2}\log(2\pi). \tag{2.2.9}$$

Kernel hyperparameter θ can be obtained by following optimization

$$\hat{\theta} = arg\max_{\theta} \log p(Y|X,\theta), \tag{2.2.10}$$

which can be solved by gradient-based optimization methods. Generally, there are many local minima.

Prediction

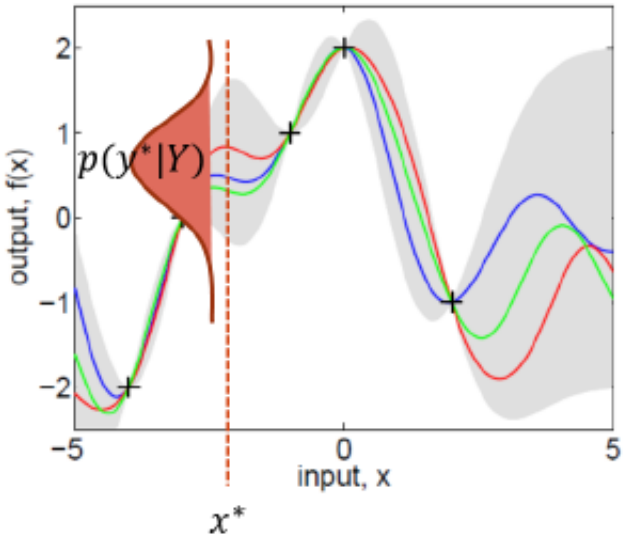


Figure 2.2: New output prediction using Gaussian process regression. [2]

Once the kernel hyperparameter θ is determined by learning, for new input $\mathbf{x}^*$, new observation y^* has a Gaussian distribution. The probability distribution for y^* can increase the reliability of the estimation by expressing them as conditional probabilities for given data and its form is the same as the conditional probability distribution of Gaussian distribution and has the following equation

$$y^* \sim \mathcal{N}\left(\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)\right) \tag{2.2.11}$$

$$\mu(\mathbf{x}) = Y^T K^{-1} k(\mathbf{x}) \tag{2.2.12}$$

$$\sigma^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - k^T(\mathbf{x}) K^{-1} k(\mathbf{x}). \tag{2.2.13}$$

In the above equation, $\mu(\mathbf{x}), \sigma^2(\mathbf{x})$ are mean and covariance respectively, and $k(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}), \cdots, k(\mathbf{x}_N, \mathbf{x})]^T \in \mathbb{R}^N$.

[2] C. E. Rasmussen, *Gaussian processes for machine learning*, 2006.

2.2.2 Gaussian Process Latent Variable models

The Gaussian Process Latent Variable Model (GPLVM)[24] can be viewed as a kind of non-linear dimensionality reduction methods using the Gaussian process, It is a type of machine learning technique that expresses a non-linear mapping between given data and its low-dimensional variables (or latent variables) as probability density function.

Data such as joint trajectory or robots, image sequence, etc. resides in very high dimensional space and high-dimensionality causes many problems in optimization, motion planning, machine learning and so on. Algorithms such as Principal Component Analysis (PCA)[25], Locally Linear Embedding (LLE)[26], and Isomap[27] have been studied to solve these problems by embedding high dimensional data in low-dimensional space. GPLVM is a method of modeling observation data into a Gaussian process and finding latent space and latent variables and has been applied to motion generation based on style-specific inverse kinematics[28], a method of naturally generating the motion of the animation characters from the motion capture data of a person[29], and etc.

For a given training data $\{y_i\}_1^N, y_i \in \mathbb{R}^D$, it can be modeled as a Gaussian process using the following equation:

$$\begin{aligned} y_i &= f(x_i) \qquad (2.2.14) \\ &= [f_1(x_i), f_2(x_i), \cdots, f_D(x_i)]^T \\ f_k &= GP(m(x_i), k(x_i, x_j)) \quad i, j = 1, \cdots, N \qquad (2.2.15) \end{aligned}$$

where $m(x)$ is a mean function, $k(x, x')$ is a kernel function, and low-dimensional latent variables $\{x_i\}_1^N$, $x_i \in \mathbb{R}^d$, $D > d$ corresponding to training data are unknowns. For the sake of simplicity, it is assumed that the mean of given data

$\mu \in \mathbb{R}^D$ is deducted from the data, and in this case, the mean function of the Gaussian process $m(x)$ is 0. As mentioned in Section 2.2.1, the characteristics of the Gaussian process depends on how you define the kernel function. In previous papers on motion data learning, RBF kernels have been most widely used and are defined as follows:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \alpha_1 \exp\left(-\frac{\alpha_2}{2}||\mathbf{x}_i - \mathbf{x}_j||^2\right) + \alpha_3 \delta_{i,j}, \tag{2.2.16}$$

where $\alpha = [\alpha_1, \alpha_2, \alpha_3]$ is kernel hyperparameter and $\delta_{i,j}$ is Kronecker delta function.

If the given data is expressed in a matrix form as $Y = [Y_1, Y_2, \cdots, Y_D] \in \mathbb{R}^{N \times D}$, then $Y_k = [y_{1k}, y_{2k}, \cdots, y_{Nk}]^T \in \mathbb{R}^N$ which is a collection of data corresponding to the k-th dimension of the given data. Conditional probability of training data Y given latent variables $X = [x_1, x_2, \cdots, x_N]^T \in \mathbb{R}^{N \times d}$ and hyperparameter α is expressed as follows:

$$\begin{aligned} p(Y_k|X, \alpha) &= \mathcal{N}(0, K) && (2.2.17)\\ &= \frac{1}{\sqrt{(2\pi)^N |K|}} \exp\left(-\frac{1}{2} Y_k^T K^{-1} Y_k\right) && (2.2.18)\\ p(Y|X, \alpha) &= \prod_1^D p(Y_k|X, \alpha) && (2.2.19)\\ &= \frac{1}{\sqrt{(2\pi)^{ND} |K|^D}} \exp\left(-\frac{1}{2} tr\left(K^{-1} Y Y^T\right)\right) && (2.2.20) \end{aligned}$$

where K is a kernel matrix whose elements $K_{i,j}$ are defined by kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$.

Learning

Learning GPLVM is to find latent variables X corresponding to given training data Y and kernel hyperparameter α. Learning is done by optimizing log posterior distribution $\log p(X, \alpha|Y)$. That is, it can be obtained by maximizing the following equation

$$\begin{aligned} \min_{X,\alpha} L &= -\log p(X, \alpha|Y) && (2.2.21) \\ &= -\log p(Y|X, \alpha)p(X)p(\alpha). && (2.2.22) \end{aligned}$$

In the above equation, the prior distribution for X assumes that

$$p(X) = \prod_{1}^{D} \mathcal{N}(0, I_{N\times N}). \tag{2.2.23}$$

To solve this optimization problem, we use the Scaled Conjugate Gradient (SCG) algorithm proposed in the previous paper. By solving this optimization problem, we can obtain latent variables X corresponding to training data Y, kernel hyperparameter α, and a smooth mapping from latent space to given data space.

New Pose Generation

After obtaining latent variables X corresponding to training data Y and kernel hyperparameter α, probability density function of new observation y for the new latent variable x is given as:

$$\begin{aligned} p(y|x, X, \alpha) &= \mathcal{N}\left(\mu(x), \sigma^2(x) I_{D\times D}\right) && (2.2.24) \\ \mu(x) &= Y^T K^{-1} k(x) && (2.2.25) \\ \sigma^2(x) &= k(x, x) - k^T(x) K^{-1} k(x), && (2.2.26) \end{aligned}$$

where $k(x) = [k(x_1, x), k(x_2, x), \cdots, k(x_N, x)]^T \in \mathbb{R}^N$ and $I_{D\times D}$ is $D \times D$ identity matrix. As in GPR, $\mu(x)$ and $\sigma^2(x)$ have the same shape as the mean and variance

of the conditional probability distribution of Gaussian distribution, respectively. The variance $\sigma^2(x)$ is a value that reflects the uncertainty in reconstruction from latent space to data space.

2.2.3 Gaussian Process Dynamical Models

The Gaussian Process Dynamical Model (GPDM)[1, 2] is an extension of GPLVM. GPDM is a model that has been studied to reflect the time series information contained in observation data into GPLVM. Unlike GPLVM, GPDM gives dynamical prior to latent variables. A common way to incorporate time series information into a model is to assume that the dynamics is parametrized time series information and determine the parameters that best fit the given data using optimization. In these model identification studies, there are many difficulties on the parameterized model (1) when the data is limited, (2) when the model is complex, and (3) when the number of parameters is high. One of the advantages of GPLVM / GPDM is that it can greatly reduce the burden of the algorithm finding many parameters (infinite parameters in RBF kernel case) by assuming them as Gaussian random variables and marginalizing these parameters.

Unlike GPLVM, GPDM finds latent variables through learning process by assigning dynamical prior to latent variables, obtains a nonlinear mapping between given data and latent variables as a closed-form probability density function, and also obtains the dynamical model between adjacent latent variables as a closed-form probability density function. These two functions can be represented by only a handful of hyperparameters that represent kernel functions by marginalizing many parameters.

For given training data $Y = \{y_i\}_1^T, y_i \in \mathbb{R}^D$ and corresponding latent variables

$X = \{x_i\}_1^T, x_i \in \mathbb{R}^d$, assuming the stochastic Markov dynamics process in the low-dimensional space $\mathbb{R}^d$, we can model the mappings as

$$x_t = f(x_{t-1}) + n_{x,t} \tag{2.2.27}$$

$$y_t = g(x_t) + n_{y,t} \tag{2.2.28}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$, $g : \mathbb{R}^d \to \mathbb{R}^D$ are nonlinear functions, $n_{x,t}, n_{y,t}$ are zero-mean isotropic Gaussian white noise process, and latent variables X are unknown. In the GPDM framework, the above two functions f and g can be expressed as a linear combination of the scalar basis functions $\phi(x), \psi(x) \in \mathbb{R}$:

$$f(x) = \sum_i^m a_i \phi_i(x) \tag{2.2.29}$$

$$g(x) = \sum_j^l b_j \psi_j(x) \tag{2.2.30}$$

where $A = [a_1, a_2, \cdots, a_m]^T \in \mathbb{R}^{m \times d}$, $B = [b_1, b_2, \cdots, b_l]^T \in \mathbb{R}^{l \times D}$. Assuming that each column of $AandB$ is isotropic Gaussian prior, conditional probability density of the data Y and the latent variable X can be obtained. First, if the parameter B is marginalized for the function g, then the conditional probability density of the data Y given X and $\bar{\beta}$ is

$$p(Y|X, \bar{\beta}, W) = \frac{|W|^N}{\sqrt{(2\pi)^{ND}|K_Y|^D}} \exp\left(-\frac{1}{2}\mathrm{tr}(K_Y^{-1} Y W^2 Y^T)\right), \tag{2.2.31}$$

where $\bar{\beta}$ is kernel hyperparameter and depends on how we define the kernel. W is a weight matrix, $K_Y \in \mathbb{R}^{N \times N}$ is a kernel matrix whose elements are defined by following RBF kernel:

$$k_Y(x, x') = \beta_1 \exp\left(-\frac{\beta_2}{2}||x - x'||^2\right) + \beta_3 \delta_{x,x'} \tag{2.2.32}$$

where $\bar{\beta} = \{\beta_1, \beta_2, \beta_3\}$ and $\delta_{x,x'}$ is Kronecker delta function.

Similarly, marginalizing the parameter A for the function f results in the conditional probability density of data X as

$$p(X|\bar{\alpha}) = \frac{p(x_1)}{\sqrt{(2\pi)^{(N-1)d}|K_X|^d}} \exp\left(-\frac{1}{2}\mathrm{tr}(K_X^{-1}X_{out}X_{out}^T)\right), \tag{2.2.33}$$

where $\bar{\alpha}$ is kernel hyperparameter and $X_{out} = \{x_t\}_2^N$ are all latent variables except x_1. Therefore, the kernel matrix is defined as $K_X \in \mathbb{R}^{(N-1)\times(N-1)}$ which consists of $X_{in} = \{x_t\}_1^{N-1}$. The kernel function that reflects the dynamical characteristics of time series data is a linear kernel combined with the RBF kernel and can be defined as follows:

$$k_X(x, x') = \alpha_1 \exp\left(-\frac{\alpha_2}{2}||x - x'||^2\right) + \alpha_3 x^T x' + \alpha_4 \delta_{x,x'} \tag{2.2.34}$$

where $\bar{\alpha} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ are kernel hyperparameters.

Learning

Learning GPDM is to find latent variables X corresponding to given training data Y and kernel hyperparameter $\bar{\alpha}, \bar{\beta}$. Learning is to find $\{X, \bar{\alpha}, \bar{\beta}\}$ that maximizes posterior distribution $p(X, \bar{\alpha}, \bar{\beta}|Y)$. This is equivalent to minimizing negative log-posterior. In other words,

$$\begin{aligned} \min_{X,\bar{\alpha},\bar{\beta}} \mathcal{L} &= -\log p(X, \bar{\alpha}, \bar{\beta}|Y) \\ &= \mathcal{L}_Y + \mathcal{L}_X + \sum_i \ln \alpha_i + \sum_j \ln \beta_j \end{aligned} \tag{2.2.35}$$

where

$$\begin{aligned} \mathcal{L}_X &= \frac{d}{2}\ln|K_X| + \frac{1}{2}\mathrm{tr}\left(x_1^T x_1\right) \\ \mathcal{L}_Y &= \frac{D}{2}\ln|K_Y| + \frac{1}{2}\mathrm{tr}\left(K_Y^{-1}YW^2Y^T\right) - N\ln|W|. \end{aligned}$$

As in GPLVM, we use the SCG algorithm to optimize the above objective function.

New Pose Generation

Once the model parameter β and the latent variable X have been determined, the new observational space coordinate y^* for the new latent space coordinate x^* is given by the following probability density function:

$$y^* \sim \mathcal{N}\left(\mu_Y(x^*), \sigma_Y^2(x^*) I_D\right) \tag{2.2.36}$$

$$\mu_Y(x) = Y^T K_Y^{-1} k_Y(x) \tag{2.2.37}$$

$$\sigma_Y^2(x) = k_Y(x,x) - k_Y^T(x) K_Y^{-1} k_Y(x) \tag{2.2.38}$$

where $k_Y(x) = [k_Y(x_1, x), k_Y(x_2, x), \cdots, k_Y(x_N, x)]^T \in \mathbb{R}^N$ and I_D is D-dimensional identity matrix. The mean function $\mu_Y(x)$ is the mean of the GP for pose reconstruction as a function of the latent space position x. The variance $\sigma_Y^2(x)$ is a value that reflects the uncertainty in reconstruction from latent space to data space.

Propagation in Latent Space

If latent variables X and dynamical model parameter $\bar{\alpha}$ in latent space are determined, for new latent coordinate x_t^*, we can obtain a probability distribution for the next probable coordinate x_{t+1}^*. A new motion can be generated from this probability distribution.

$$x_{t+1}^* \sim \mathcal{N}\left(\mu_X(x_t^*), \sigma_X^2(x_t^*) I_d\right) \tag{2.2.39}$$

$$\mu_X(x) = X_{out}^T K_X^{-1} k_X(x) \tag{2.2.40}$$

$$\sigma_X^2(x) = k_X(x,x) - k_X^T(x) K_X^{-1} k_X(x) \tag{2.2.41}$$

where $k_X(x) = [k_X(x_1, x), k_X(x_2, x), \cdots, k_X(x_{N-1}, x)]^T \in \mathbb{R}^{N-1}$, I_d is d-dimensional

identity matrix and $X_{out} = \{x_t\}_2^N$ for the first-order Markov process assumption. The mean function $\mu(x)$ is the function that represents the next most likely coordinate from the current latent space coordinate x. The variance $\sigma_X^2(x)$ is a measure of the uncertainty of the predicted coordinates from the current coordinate x.

2.3 Forward Kinematics of Open Chains

The Special Orthogonal Group

The Special Orthogonal Group $SO(3)$ is defined as follows:

$$SO(3) = \{R \in \mathbb{R}^{3\times3} | RR^T = R^T R = I, \det(R) = +1\}. \tag{2.3.42}$$

The Lie algebra of $SO(3)$, denoted $so(3)$, consists of the real $n \times n$ skew-symmetric matrices:

$$so(3) = \{[\omega] = \mathbb{R}^{3\times3} | [\omega]^T = -[\omega], \omega \in \mathbb{R}^3\}, \tag{2.3.43}$$

the square bracket notation $[\omega]$ defined as the following 3×3 skew-symmetric matrix representation of ω:

$$[\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{2.3.44}$$

The Special Euclidean Group

The Special Euclidean Group $SE(3)$ is defined as follows:

$$SE(3) = \{M \in \mathbb{R}^{4\times4} | M = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}, R \in SO(3), p \in \mathbb{R}^3\}, \tag{2.3.45}$$

,for convenience we will also use the more compact notation $M = (R, p) \in SE(3)$.

The Lie algebra of $SE(3)$, denoted $se(3)$, is of the form

$$[\mathcal{S}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix}, \tag{2.3.46}$$

where $[\omega] \in so(3)$ and $v \in \mathbb{R}^3$. $[\mathcal{S}]$ will be written in the equivalent six-dimensional column vector form as $\mathcal{S} \in \mathbb{R}^6$, and also using the notation $\mathcal{S} = (\omega, v) \in \mathbb{R}^6$.

Product of Exponentials Formula

Consider an m-degree of freedom open chain whose forward kinematics is represented in the following product of exponentials form [17, 30, 31]:

$$f(\theta) = e^{[\mathcal{S}_1]\theta_1} \cdots e^{[\mathcal{S}_m]\theta_m} M, \tag{2.3.47}$$

where θ_i, $i = 1, \ldots, m$ denote the joint variables, and $M \in SE(3)$ and $[\mathcal{S}_i] \in se(3)$ are of the form

$$M = \begin{bmatrix} R_M & p_M \\ 0 & 1 \end{bmatrix}, \quad [\mathcal{S}_i] = \begin{bmatrix} [\omega_i] & v_i \\ 0 & 0 \end{bmatrix}, \tag{2.3.48}$$

with $R_M \in SO(3)$, $p_M \in \mathbb{R}^3$.

Adjoint Map and Jacobian

For any $X = (R, p) \in SE(3)$, the 6×6 adjoint matrix $[\mathrm{Ad}_X]$ is defined as follows:

$$[\mathrm{Ad}_X] = \begin{bmatrix} R & 0 \\ [p]\,R & R \end{bmatrix}. \tag{2.3.49}$$

Letting $p \in \mathbb{R}^3$ be the Cartesian position of the end-effector frame expressed in fixed frame coordinates, its velocity $\dot{p}$ is then given by

$$\dot{p} = \omega_s \times p + v_s, \tag{2.3.50}$$

where

$$\begin{bmatrix} \omega_s \\ v_s \end{bmatrix} = J_s(\theta)\dot{\theta} = \begin{bmatrix} \mathcal{S}'_1 & \cdots & \mathcal{S}'_m \end{bmatrix} \dot{\theta} \tag{2.3.51}$$

$$\mathcal{S}'_i = \left[\mathrm{Ad}_{e^{[\mathcal{S}_1]\theta_1 \ldots e^{[\mathcal{S}_{i-1}]\theta_{i-1}}}}\right] \mathcal{S}_i, \tag{2.3.52}$$

with the adjoint matrix $[\mathrm{Ad}_{(\cdot)}]$ as defined in (2.3.49). Each of the six-dimensional vectors $\mathcal{S}'_i = (\omega'_i, v'_i)$ depends explicitly on θ, and represents the screw motion corresponding to the i-th joint axis when the robot is at the configuration θ.

3

GPDM-Based Human Motion Intention Recognition for Lower-Limb Exoskeleton

3.1 Introduction

Exoskeleton robots have been studied for military purposes, work assistance in industrial settings, and assisting people with disabilities or needing rehabilitation [32, 33]. In order to effectively assist the human movement, the exoskeleton robot should effectively recognize the human motion intention and apply the appropriate control algorithm. Studies to recognize or predict human motion intentions have relied primarily on electromyography (EMG) signals [3], [4], [5]. EMG signals can be acquired immediately before the muscle is activated, and thus it is possible to predict the user's motion intention quickly. On the other hand, it is

necessary to calibrate the sensor value before using the EMG signal because the intensity of the signal is different for each person and also depending on the attachment position of the EMG sensor and the condition of the user [34, 3, 35]. This characteristic is ineffective in the practical use of the exoskeleton robot, and can hardly be used especially for military purposes. In order to solve these drawbacks, research [6] has been conducted to capture the motion intention without using EMG signals. In that study, using joint angle, joint angular velocity, and joint torque data, they studied three different movement types using Linear Discriminant Analysis (LDA), which is one of the machine learning techniques. However, in order to describe a person's intentions, three are not enough and the LDA is a machine learning algorithm that cannot reflect the dynamic characteristics of the motion. In [7], they used Support Vector Machine (SVM) to identify five types of movement states: walking on a flat ground, walking up and down the stairs, and walking up and down the incline. Including sensor information such as EMG sensor, pressure sensor, and joint angle, the accuracy of motion recognition of 99% was shown. However, SVM is also a machine learning algorithm that does not reflect the dynamic characteristics of motion. In our preliminary study analysis without using the EMG sensor, we showed very low motion recognition rate using SVM.

In this chapter, we applied the Gaussian process dynamical model (GPDM) [1, 2], which is a nonlinear machine learning method. GPDM can reflect the dynamic characteristics of the exercise data to recognize the intentions of the wearer of exoskeleton robot. GPDM is able to express human motion efficiently even in three dimensions and has been used in fields such as human tracking[36], animation graphics[37], and so on. To verify the algorithm, the algorithm was applied to the human motion data and the exoskeletal robot measurement data. Then we

compared ours with the standard continuous HMM[38]-based motion recognition algorithm.

The remainder of this chapter is organized as follows. The Section 3.2 introduces our motion intention recognition algorithm using GPDM. Section 3.3 describes two kinds of data used in the experiments of this chapter. Section 5.3 presents experimental results using our method and an HMM-based method. Discussion about experimental results are made in Section 3.5. Then conclusion remarks are described in Section 5.4.

3.2 Human Motion Intention Recognition using GPDM

By learning GPDM K times for given K different motion data $Y_k \in \mathbb{R}^{N_k \times D}$, $k = 1, \cdots, K$, a low dimensional variable (latent variable) X_k and a parameter set $\lambda_k = \{\bar{\alpha}_k, \bar{\beta}_k, W_k\}$ can be obtained. Given the new observation data Y^*, the latent variable X_k^* for each motion model can be obtained through Gaussian process regression (GPR). The motion intention estimation of the new observation data Y^* can be estimated by comparing the following conditional probability distributions

$$\mathcal{L}_k = p(Y^*|Y_k, X_k, \lambda_k)p(X^*|X_k, \lambda_k) \tag{3.2.1}$$

and choosing the model k, which represents the greatest probability. That is,

$$\lambda^* = arg\max_{\lambda_k} \log \mathcal{L}_k \tag{3.2.2}$$

$$= arg\max_{\lambda_k} \log \left(p(Y^*|Y_k, X_k, \lambda_k)p(X^*|X_k, \lambda_k)\right). \tag{3.2.3}$$

By selecting the model λ_k corresponding to $\mathcal{L}_k$, which has the maximum value in the above equation, the current motion corresponding to the current data can be estimated.

3.3 Data Description

There are two main datasets used in this chapter: the human motion data and the sensor mock-up data. The former is a data that is obtained by attaching markers to each part of the body, tracking the position of the three-dimensional markers using the 3D motion capture system, and then converting them into the joint angle values using the dynamics simulation information. The latter is the data collected after a human being actually wears the sensor mock-up (lower-limb exoskeleton robot with sensors only, without actuator). On the flat, stair, and slope modeling the actual operating environment of the robot, walking, sitting and standing, etc. were performed. Each data consists of various sensor information such as joint angle, IMU (Inertial Measurement Unit) and AHRS (Attitude and Heading Reference System).

3.3.1 Human Motion Capture Data

Before making the Lower-Limb Exoskeleton Robot, human body kinetic analysis data was collected to analyze the human kinetic performance, and its characteristics were reflected in the robot design. This data is based on motion capture and ground reaction force measurements for all 12 subjects. The average height of the subjects was $172.3 \pm 5.4cm$ and the average weight was $71.35 \pm 6.18kg$. Motion capture is done by the VICON motion capture camera system. The detailed specification of the motion capture system is summarized in the following table 3.1.

The position of the markers attached to the human body for motion capture is shown in the following Figure 3.1.

The ground reaction force was measured by the AMTI ground reaction force

Table 3.1: VICON motion capture camera system

Model	VICON MX-T160
SW	VICON Nexus 1.8
Capture rate	100Hz
Marker Set	VICON Plug-in Gait

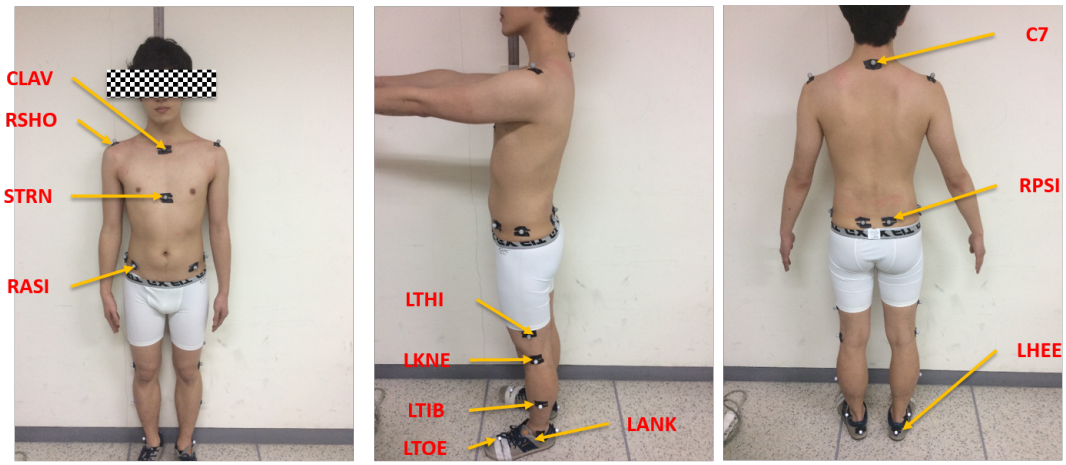


Figure 3.1: Marker positions for motion capture system.

measurement system, and the detailed specifications are shown in the table 3.2. The motion capture data representing the three-dimensional position of the markers was converted into joint angles information through dynamics simulation in combination with information such as human body model and ground reaction force. The joint angle information is composed of 30 dimensions in total.

Using the above measurement system, twelve subjects performed six actions (walking on flat ground, walking across the incline, walking up/down the incline, walking up/down the stairs) three times each. The experimental environment in

Table 3.2: AMTI ground reaction force measurement system

Model	AMTI-OR6-7-2000
Capture rate	1000Hz

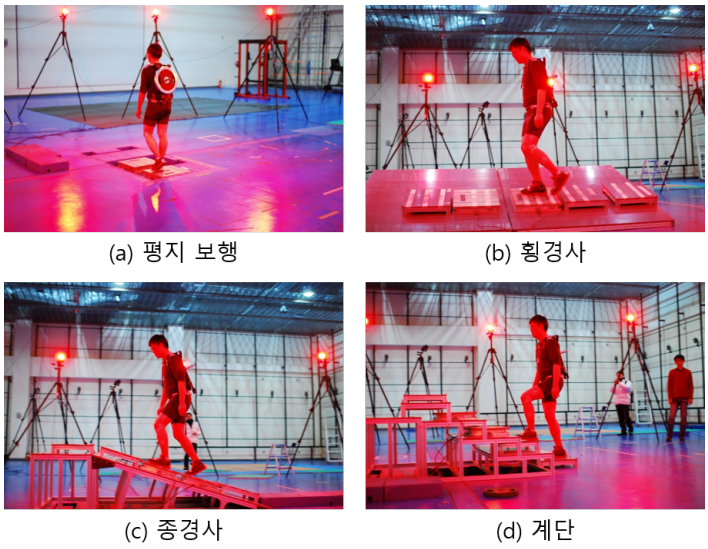


Figure 3.2: The experimental environment for human kinetic analysis.

which each operation is performed can be seen in the Figure 3.2.

Each measured gait data was normalized to 100 frames for 1 gait cycle, beginning at one foot of the heel strike (HS) and ending up to the foot HS. The terminology and sequence for the walking cycle are described in Figure 3.3.

Human kinetic analysis data information is summarized in the table 3.3.

Table 3.3: Human Motion Capture Data

Item	Value	note
subject	12 person	Height: $172.3 \pm 5.4cm$ Weight: $71.35 \pm 6.18kg$
movement types	6 types	walking on flat ground, walking across the incline, walking up/down the incline, walking up/down the stairs
frames	100 frame	1 gait cycle
sets per movement type	3 sets	
data dimension	30 dimension	joint angles

3.3.2 Sensor Mock-up Data

Sensor mock-up data is a collection of data that subjects, wearing lower-limb exoskeleton robot with sensors only (actuators are not attached), walk on terrain that models the actual operating environment of the robot, such as flat ground, obstacles, stairs, and inclines. These data were obtained from four subjects and sensor information such as joint angle, IMU, and AHRS were collected. The average height of the subjects was $173.7 \pm 2.7cm$ and the average weight was $74.58 \pm 5.18kg$. The joint structure of sensor mock-up is shown in Figure 3.4. The hip joint consists of five revolute joints, the ankle joint has three revolute joints, and the knee has one revolute joint. It has a total of 18 joints and the total weight is $6.5kg$. Each joint value is measured by an encoder. The AHRS sensor is attached to the back plate. The IMU sensors are attached to both feet. Two values of roll and

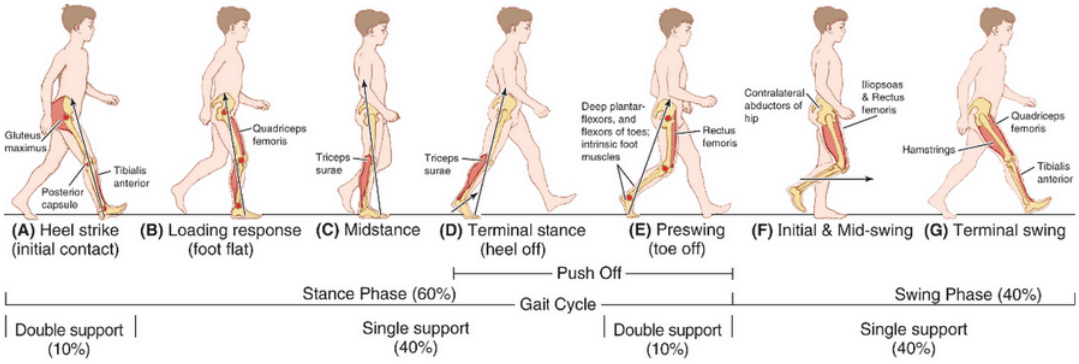


Figure 3.3: Gait cycle. [1]

pitch were used in the AHRS sensor values except for the yaw value. In the IMU sensor, the values of 3-axis linear acceleration and 3-axis orientation are obtained, so 12-dimensional data is used as a result. There is a total of 32 data dimensions including 18 joints, 2 in AHRS, and 12 in IMU sensors.

All measurements values were measured at every $16ms$ and the frame rate was $62.5Hz$.

Wearing the above sensor mock-up, four subjects performed nine motions: walking on flat ground, walking across the incline, walking up/down the incline, walking up/down the stairs, squat, walking over an obstacle, and walking sideways three times each. The experimental environment in which each movement is performed can be seen in the Figure 3.5.

The experimental data that subjects carried loads on their back were also measured, but they were not used in this study. Unlike the human kinetic analysis data, it was not normalized to the gait cycle and was recorded at a certain time

[1] http://epomedicine.com/clinical-medicine/physical-examination-gait/

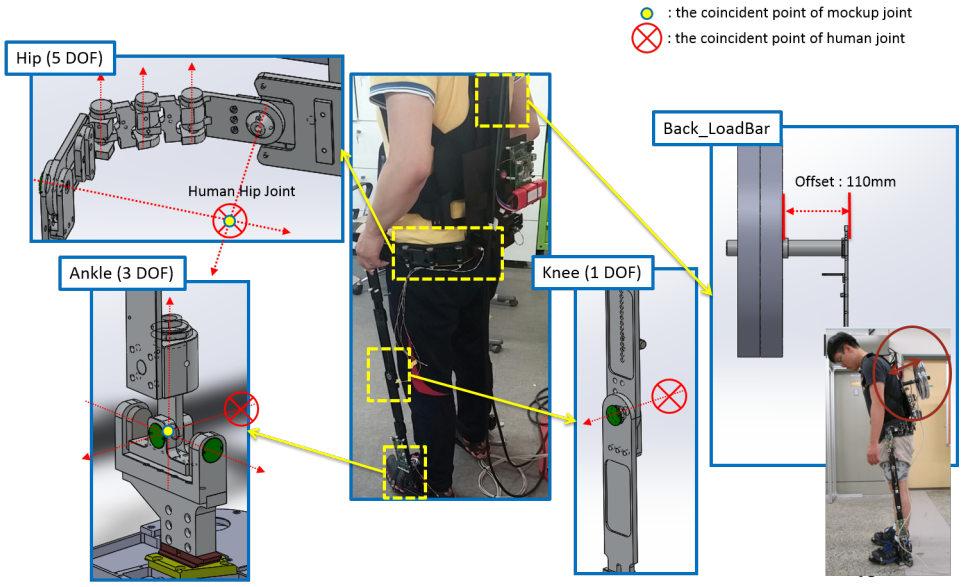


Figure 3.4: Sensor mockup

interval according to the frame rate. Data was recorded during 1 to 4 gait cycles according to the movement types. Walking on flat ground was measured over the longest period and walking over an obstacle was measured for one gait cycle. For walking on flat ground, the walking speed was about $2km/h$ according to the metronome, and other movements were measured freely. Sensor mock-up data information are listed in the Table 3.4.

3.4 Experiments

3.4.1 Previous Research

In this section, we examined whether the SVM algorithm, which was used in previous study [7] to identify the motion intention of the wearer of the prosthetic

Figure 3.5: Sensor mock-up experiment environment.

leg, can be applied to data given without EMG signals. In the previous study [7], including the EMG signals, the force and moment at the sole from the load cell and pressure at the contact area with the prosthesis were measured. The pressure of the sole was used to divide the gait cycle into four steps (combinations of swing phase and stance phase for both feet). Based on the EMG, force, and moment values for each step, they performed a study of learning and classifying a total of 5 movements (walking on flat ground, walking up/down the incline, walking up/down the stairs) using SVM. In this study, classification accuracy was $91.79 \sim 100\%$ for each movement type.

In order to verify the above data without EMG signal, the SVM algorithm was applied to sensor mock-up data. The Figure 3.6 is a confusion matrix that indicates how accurate the class is classified. In the Figure 3.6, the vertical axis represents the actual movement type and the horizontal axis represents the predicted movement type. The numbers in the confusion matrix have $0 \sim 1$ values and indicate how the actual movement type is classified by the movement type

Table 3.4: Sensor Mockup Data

Item	Value	Note
Subject	4 person	height: $173.7 \pm 2.7cm$ weight: $74.58 \pm 5.18kg$
movement types	9 types	walking on flat ground, walking across the incline, walking up/down the incline, walking up/down the stairs, squat, walking over a obstacle, and walking sideways
frames	500 ∼1000 frame	$1 \sim 4$ gait cycles
sets per movement type	3 sets	
data dimension	32 dimension	joint angle, IMU, AHRS

of the horizontal axis. The closer the value is to 1, the whiter the color becomes. The closer to 0, the blacker. For example, walking up the stairs was classified as walking on flat ground with an accuracy of 59.97% and as walking up the stairs with an accuracy of 38.07%. Thus, the closer the diagonal color is to white, the better the motion is classified.

As shown in Figure 3.6, if there are no EMG signals, most of the data is classified as walking on flat ground and cannot be classified well by the original motion.

Based on these results, we assumed that the SVM using joint angles, IMU and AHRS values does not capture the dynamic characteristics of the data. Therefore, we considered the three consecutive data as one data and applied the SVM. That is, for existing data $y_t \in \mathbb{R}^D,\ t = 1, \cdots, N$, after concatenating three consecutive

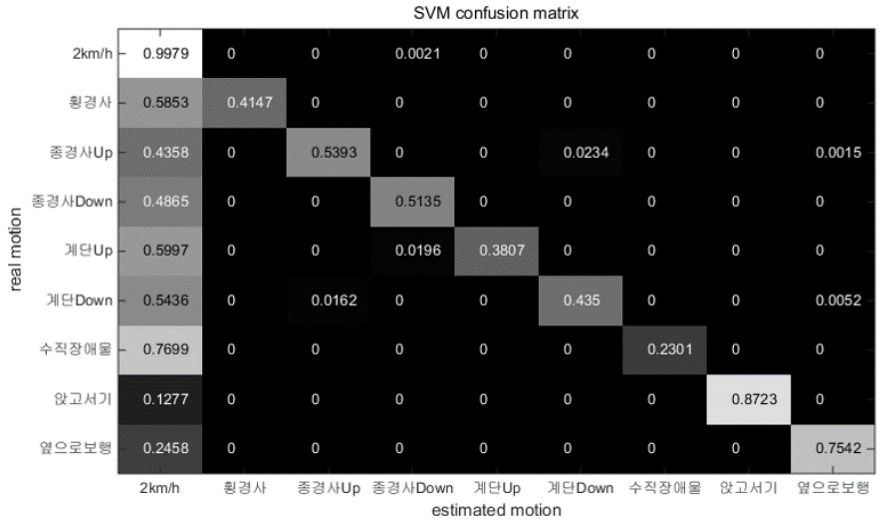


Figure 3.6: SVM without EMG

data as $\bar{y}_t = [y_t^T, y_{t+1}^T, y_{t+1}^T]^T \in \mathbb{R}^{3D},\ t = 1, \cdots, N-2$, then we applied SVM on the concatenated data $\{\bar{y}_t\}_1^{N-2}$.

As shown in Figure 3.7 confusion matrix, it can be seen that the results are worse when three consecutive data are regarded as one data. In other words, even if three consecutive data are used as one data, it can be seen that the SVM does not capture the dynamic characteristics of the data. Based on these two results, we recognized that there is a limitation of SVM when only joint angle, IMU, AHRS sensor values are available. Therefore, GPDM-based motion intention recognition that can reflect the dynamic characteristics of data is studied and compared with an HMM-based motion intention recognition method.

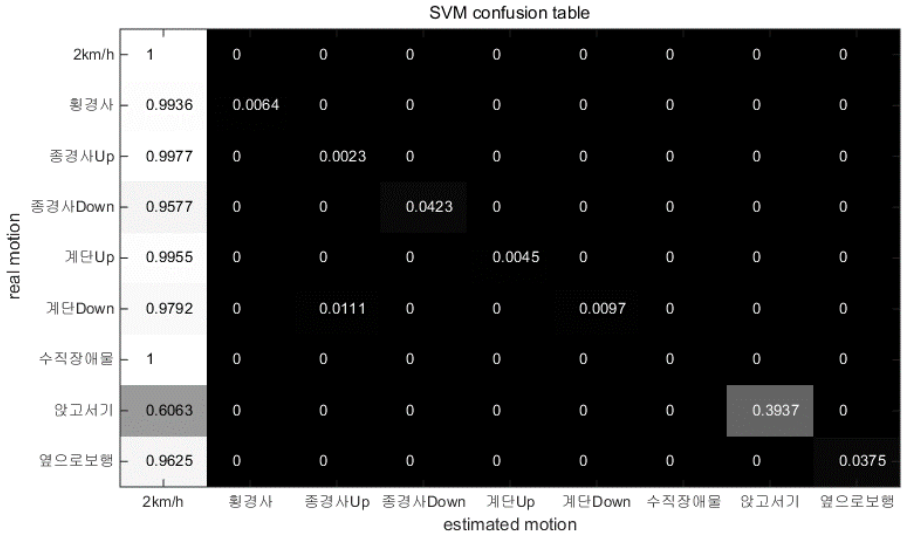


Figure 3.7: SVM without EMG. Three consecutive data are concatenated as a datum.

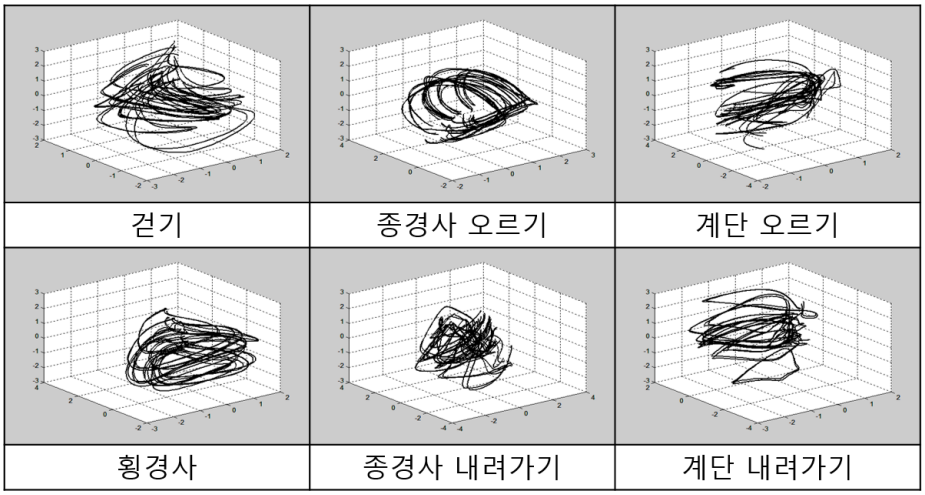


Figure 3.8: GPDM latent trajectories of human motion capture data.

3.4.2 Human Motion Capture Data

Before performing motion intention recognition using GPDM, we investigate the trajectory of low-dimensional space by learning the six motions of human kinetic analysis data using GPDM in three - dimensional space. As shown in Figure 3.8, each movement has learned in the same three-dimensional space but has different characteristics. Therefore, if enough frames of new observation data come in, it will be possible to distinguish motion intention by only comparing the low dimensional trajectory shape.

The experiment proceeded as follows. GPDM with 2nd-order Markov process in latent space is applied to the data of Table 3.3. Two sets of three sets of motion were used for learning and the remaining one set of motion was used as test data. Since the second-order Markov process is used, at least three frames of data are required to determine the motion intention based on the GPDM. Using the GPDM-based motion intention recognition algorithm in the previous Section 3.2,

the confusion matrix is constructed as shown in Figure 3.9. The classification accuracy for the six movement types that make up this data is $87.97 \pm 10.39\%$. The motion classification accuracy of the walking up the stairs and walking down the incline is somewhat lower, and the remaining movement types are accurate to more than about 89%. Intuitively, it is anticipated that walking up and down the incline will be similar to walking up and down the stairs, respectively. As you can see from the Figure 3.9, the walking up the stairs is highly likely to be misclassified as walking up the incline, and the walking down the stairs is highly likely to be misclassified as walking down the incline.

In order to compare with HMM-based motion recognition algorithm, a confusion matrix in Figure 3.10 is constructed by learning HMM under the same condition. For the efficiency of HMM learning, it was reduced to five-dimensional space using LDA and then learned by HMM. If the data has high dimensionality, there are many local minimums, and HMM learning is easy to fall to the local minimum because it uses the EM algorithm. Therefore, we reduce the dimensionality. LDA is one of the most basic classification algorithms in machine learning and can be reduced to a maximum of $C - 1$ dimensions for C classes. Since we have six types of motion, we have reduced the data to five dimensions. Average motion recognition accuracy is $78.75 \pm 5.74\%$, which is somewhat lower than the GPDM-based algorithm. However, similar to the GPDM-based method, we can see that the walking up the stairs is similar to the walking up the incline and down the stairs is similar to down the incline.

In order to analyze more deeply, we investigated how the motion recognition accuracy varies according to the number of frames used for motion recognition and gait cycle. The result is shown as graphs in the Figure 3.11 and the Figure 3.12. In both graphs, the vertical axis represents the motion recognition accuracy and

the horizontal axis represents the number of frames used for motion recognition. The color of the graph indicates the starting position of the data used for motion recognition in the gait cycle and represents the walking phase. We used up to 20 frames from the minimum number of frames required for motion recognition. The starting position of the data changed stepwise from the first frame to the 80th frame. The first frame (1 frame) of the data represents the HC (Heel Contact) and the 100th frame is the frame immediately before the next HC appears.

The GPDM-based algorithm showed a slight increase in classification accuracy as the number of frames used increased, but showed a tendency not to be significant. On the other hand, as the number of frames used increases, the HMM-based algorithm increases the classification accuracy and shows a maximum difference of about 25%. The HMM-based algorithm showed good performance with data close to 20 frames. On the other hand, the GPDM-based algorithm showed similar performance to the HMM-based algorithm, which uses 20 frames, with only the minimum number of frames (3 frames) required for motion recognition by the 2nd-order Markov model assumption. This means that the GPDM-based algorithm can extract the dynamic characteristics of motion with only three frame data. In case of GPDM based algorithm, the performance is good when the start position of data is $4, 80\%$ frame. This shows that the posture information before and after HC is used as a more important factor in recognizing the intention of motion than information at the stance phase of the gait cycle. For the HMM-based algorithm, the data starting from $1, 10\%$ frame showed the best performance. This shows that, unlike the GPDM-based algorithm, the posture information immediately after the HC is used as an important factor in recognizing the intention of the motion.

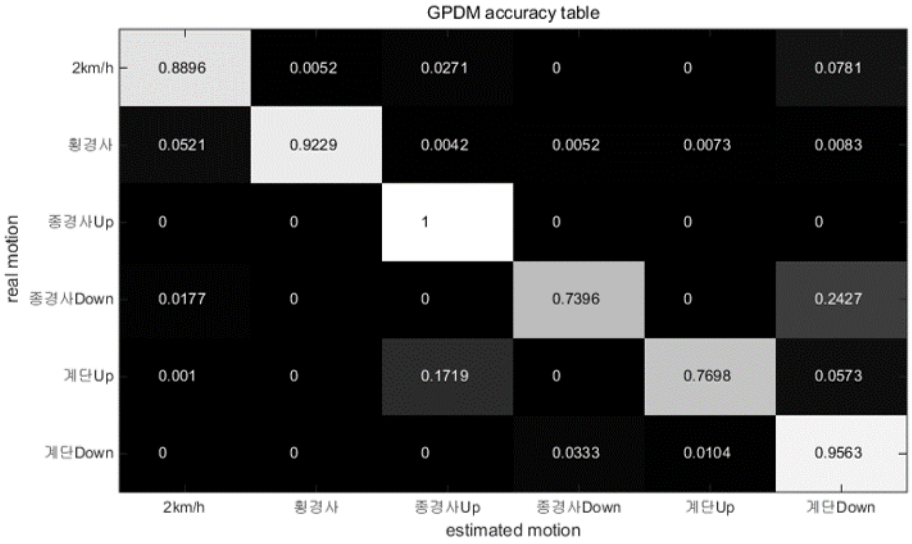


Figure 3.9: GPDM confusion matrix for human kinetic analysis data.

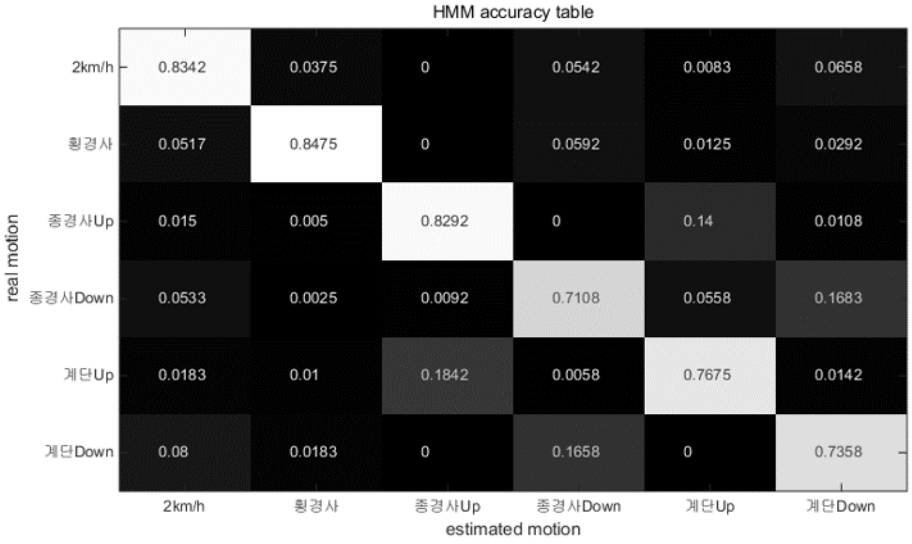


Figure 3.10: HMM confusion matrix for human kinetic analysis data.

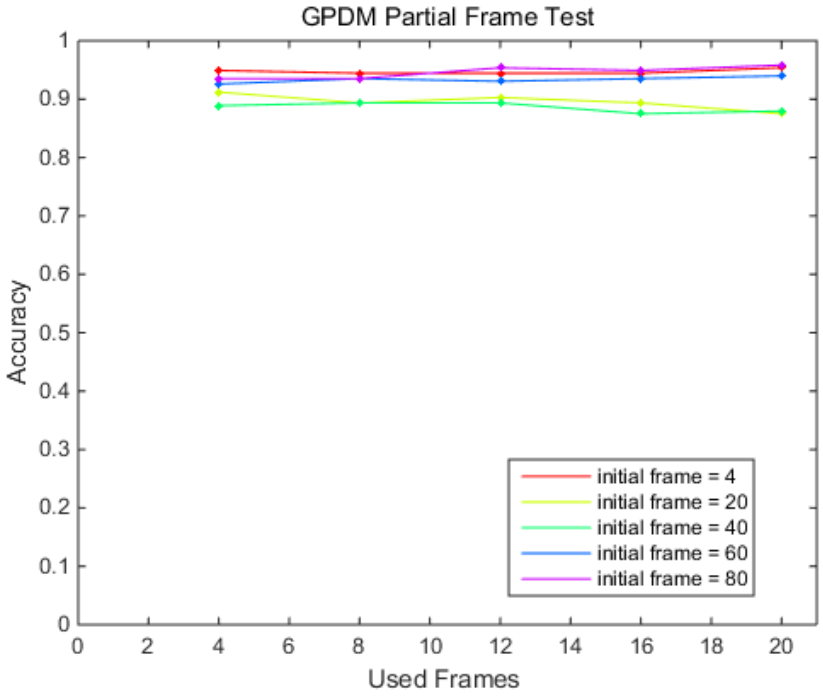


Figure 3.11: Effect of the number of frames used and gait cycle: GPDM case.

3.4.3 Sensor Mock-up Data

Based on the result of the Section 3.4.2, we applied GPDM and HMM-based motion recognition algorithm to sensor mock-up data. As before, the 2nd-order Markov model was applied to the GPDM and the HMM was applied to the 8-dimensional data whose dimensionality had been reduced using the LDA. Two sets of three sets of motion were used for learning and the remaining one set of motion was used as test data. Each classification performance for the nine movement types that constitute this data is shown as a confusion matrix in Figure 3.13 for GPDM and in Figure 3.14 for HMM, respectively. The classification accuracy for the nine movement types is $89.02 \pm 6.14\%$ for GPDM and $94.28 \pm 6.16\%$ for HMM. Whereas in

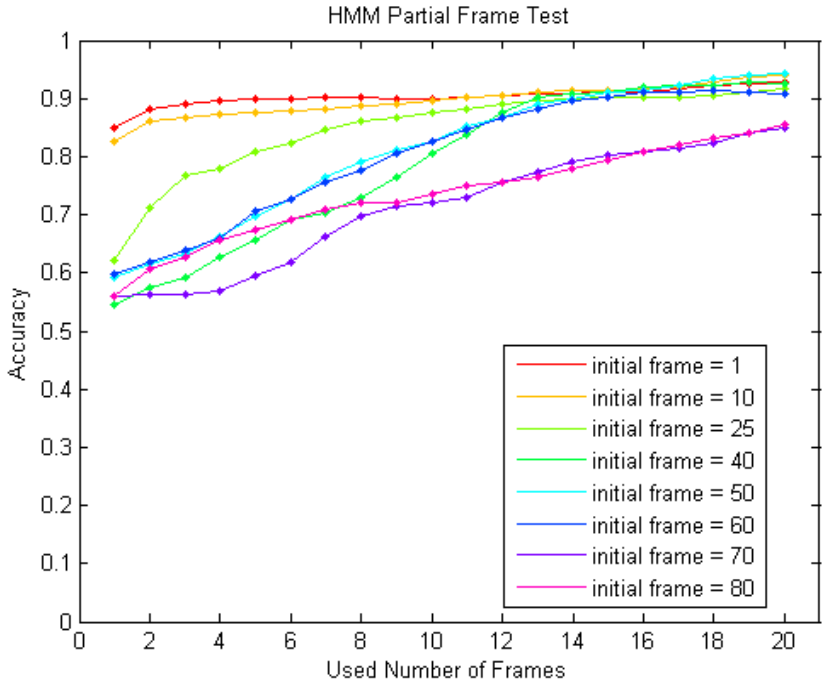


Figure 3.12: Effect of the number of frames used and gait cycle: HMM case.

the case of human motion data, the average classification accuracy of the GPDM-based algorithm was better than that of the HMM-based algorithm, in the case of sensor mock-up data, that of the HMM-based algorithm is better than that of the GPDM-based algorithm.

3.5 Discussion

3.5.1 Comparison both Data Sets

In order to analyze the causes of the two algorithms showing different results for the two data sets, the differences between the human kinetic analysis data and the

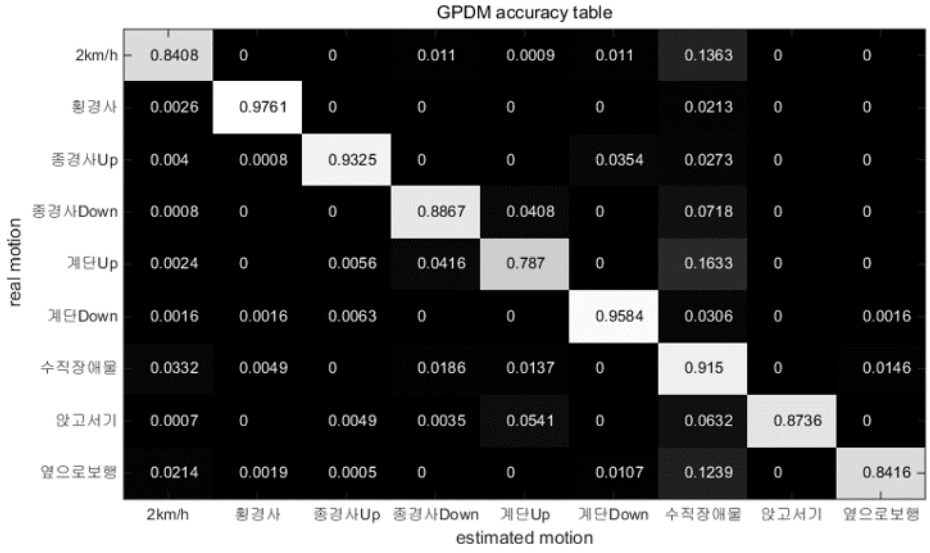


Figure 3.13: Confusion matrix of GPDM for sensor mock-up data.

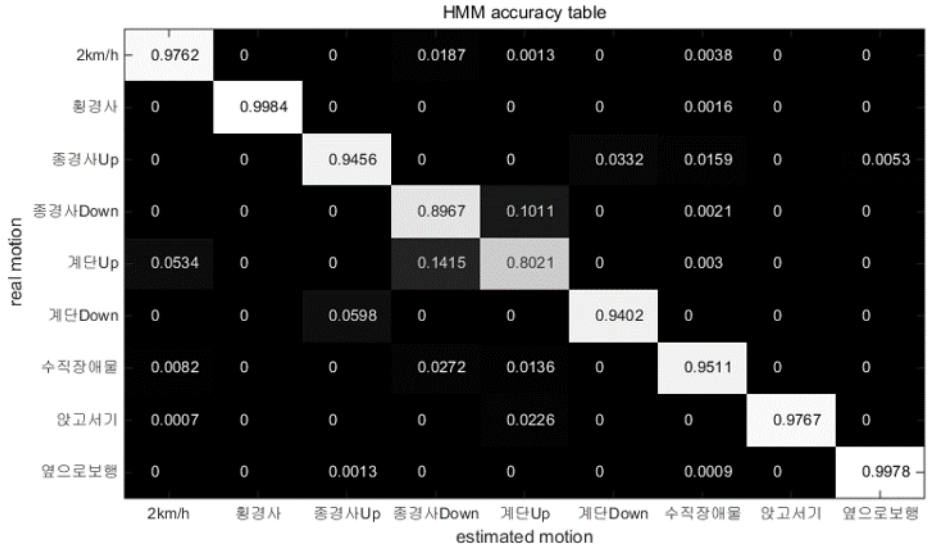


Figure 3.14: Confusion matrix of HMM for sensor mock-up data.

sensor mock-up data are compared and summarized in Table 3.5. In the case of the sensor mock-up data, the data types were more various than human kinetic analysis data and three types of motion were added, while the number of subjects participating in the sensor mock-up experiment was four, which is one-third of human kinetic analysis data. Despite the differences in data, for the GPDM-based algorithm, there is almost no difference (about 1%) in motion recognition performance between two data sets as shown in Table 3.5. On the other hand, HMM-based algorithm improves motion classification performance by about 15% for sensor mock-up data. In this sense, GPDM-based algorithms are robust to changes in data types, number of subjects, and movement types while motion classification performance of HMM-based algorithms is greatly affected by data characteristics. Especially, as the number of people increases, the noise level of the data becomes more and more diverse. Therefore, it is difficult for the machine learning algorithm to grasp the characteristics of motion. However, GPDM has the advantage that the motion classification performance is not greatly changed for the human body motion data having three times the number of people. Although the number of people has changed by three times, GPDM has shown that motion classification performance does not change much.

One of the important facts from Table 3.5 is that the dimensionality of GPDM latent space is smaller than that of the HMM latent space. When the dimensionality of latent space of GPDM is made equal to that of HMM, the result is shown as the confusion matrix of Figure 3.15. In this case, the average classification accuracy is 94.37 ± 5.46, which is similar to the HMM-based algorithm.

Analysis by each motion type shows that GPDM-based algorithm has a relatively high probability of misclassification as a vertical obstacle. This can be attributed to the fact that the motion over the vertical obstacle contains some of

Table 3.5: Data Comparison

Item	Human Motion	Sensor Mockup
Data type	Joint angle	Joint angle, IMU, AHRS
Data dimension	30	32
Number of person	12	4
Number of motion	6	9
GPDM latent dim.	3	3
HMM latent dim.	5	5
GPDM accuracy	$87.97 \pm 10.39\%$	$89.02 \pm 6.14\%$
HMM accuracy	$78.75 \pm 5.74\%$	$94.28 \pm 6.16\%$

the characteristics of all other motions. Basically, the motion over the obstacle can be regarded as an exaggerated movement of the walking on flat ground. The part that lifts the leg forward to overcome the obstacle is similar to the walking up the incline/stairs motion. The part of the leg that falls beyond the obstacle is similar to the walking down the incline/stairs. Finally, depending on the person's physical characteristics or habits, some people lift their feet sideways when crossing obstacles. Therefore, it seems that there is a possibility that the motion over the obstacle is misclassified as the sideways motion.

Despite the similarity of walking on flat ground and walking across the incline, GPDM and HMM-based algorithms well distinguished them for the two data sets. In case of walking on the incline and stairs, unlike human kinetic analysis data, both algorithms have a high probability of being confused with walking up the incline as walking down the stairs for sensor mockup data. In case of walking on the incline and stairs, unlike human kinetic analysis data, both algorithms have a

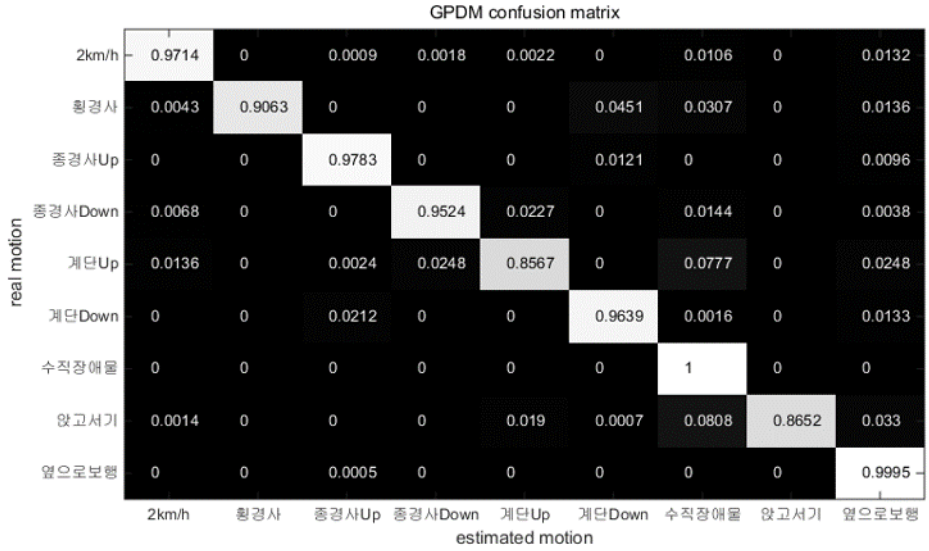
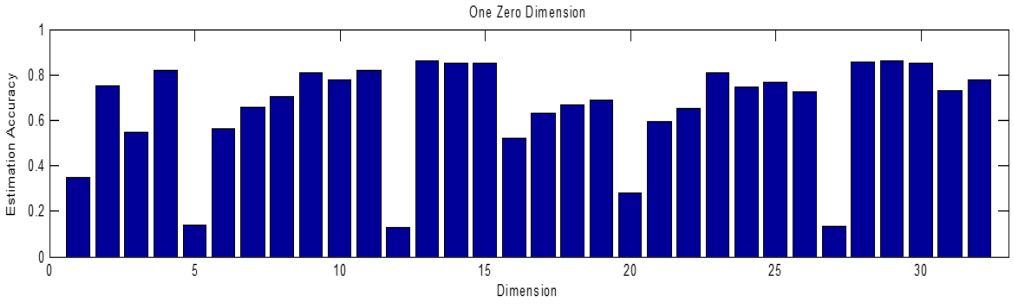


Figure 3.15: Confusion matrix of GPDM for sensor mock-up data. Dimensionality of the latent space is 5. Average classification accuracy is 94.37 ± 5.46.

high probability of being confused walking up the incline with walking down the stairs, walking down the incline with walking up the stairs, respectively for sensor mockup data. In the GPDM-based algorithm, the probability of misclassification is very low, but the probability of misclassification is high in the HMM-based algorithm. What can be inferred from this is that wearing an exoskeleton robot subtly changes the wearer's gait pattern.

3.5.2 Sensitivity Analysis

The following experiment was designed to investigate what dimension of data influences original 32-dimensional data when sensor is broken during operation of exoskeleton robot. We have examined how the motion classification performance



Dimension	1	2	3	4	5	6	7	8
Label	R Hip Angle Yaw1	R Hip Angle Yaw2	R Hip Angle Pitch	R Hip Angle Roll	R Hip Angle Yaw3	R Knee Angle Pitch	R Ankle Angle Pitch	R Ankle Angle Yaw
Dimension	9	10	11	12	13	14	15	16
Label	R Ankle Angle Roll	R Outsole Angle x	R Outsole Angle y	R Outsole Angle z	R Outsole Acc x	R Outsole Acc y	R Outsole Acc z	L Hip Angle Yaw1
Dimension	17	18	19	20	21	22	23	24
Label	L Hip Angle Yaw2	L Hip Angle Pitch	L Hip Angle Roll	L Hip Angle Yaw3	L Knee Angle Pitch	L Ankle Angle Pitch	L Ankle Angle Yaw	L Ankle Angle Roll
Dimension	25	26	27	28	29	30	31	32
Label	L Outsole Angle x	L Outsole Angle y	L Outsole Angle z	L Outsole Acc x	L Outsole Acc y	L Outsole Acc z	Body AHRS Roll	Body AHRS Pitch

Figure 3.16: Sensitivity of each dimension. (Up) Sensitivities. (Down) Labels of each dimension.

changes after setting all the data values of each dimension to zero. The results are shown in Figure 3.16. The most significant difference in performance is the 5th, 12th, and 27th dimensional data compared to the average classification accuracy of 89.02% when all the normal values are received. The labels for each dimension are Right Hip Angle Yaw 3, Right Outsole Angle Z, Left Outsole Angle Z, the sensor positions are the third yaw angle of the right hip joint, and the roll angle of the IMU sensor on the right and left.

The reason why the 5th data of the right hip joint is sensitive is that the walking across the incline movement was performed in a right-sloping environment such as Figure 3.17. The yaw angle of the right hip joint seems to play an important role in distinguishing it from other movements such as walking across the incline and walking on flat ground. On the other hand, the reason for the 12th and 27th

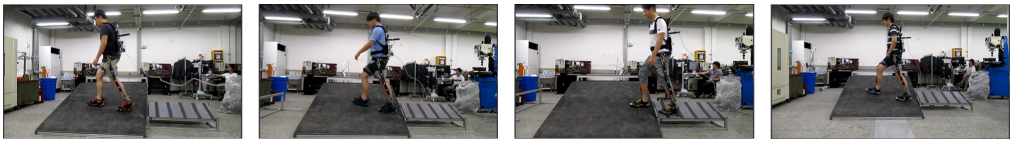


Figure 3.17: Walking across the incline. The shape of the incline is high on the right side of the subject.

data of each foot roll is sensitive seems difficult to find the cause intuitively.

3.6 Conclusion

By comparing logarithms of conditional probability distribution of new observation and the corresponding GPDM latent variables given training data of a certain motion type, we estimated motion intention of newly observed data. Using GPDM, we were successfully able to extract and capture the dynamic characteristics of movements on various terrains. Therefore we can recognize better the wearer's motion intention than an HMM-based algorithm and of course get better recognition than static algorithms such as LDA and SVM which cannot capture dynamic nature of a motion.

Our GPDM-based motion intention recognition algorithm has been validated through two kinds of data sets. The first is joint angle paths which are converted from 3D Cartesian coordinates of markers attached to subjects' body parts and regularized as 100 frames for a gait cycle. The second is data captured from various sensors attached to lower-limb exoskeleton robot. Our method accurately predicted human motion intention than an HMM-based algorithm. In addition, the performance was uniform regardless of data type, motion type, and where it was

captured during the gait cycle. This means our algorithm is more robust and reliable.

When we think intuitively, the motion of walking up the stairs will be similar to the motion of walking up the ramp, and the motion of walking down the stairs will be similar to the motion of walking down the ramp. One of the interesting things, however, is that if one wears an exoskeleton, the motion of walking up the stairs is more likely to be mistaken for a walking down the ramp and the motion of walking down the stairs is more likely to be misclassified as the motion of walking up the stairs. For motion capture data, results are consistent with our intuition. What can be inferred from this is that wearing an exoskeleton robot subtly changes the wearer's gait pattern. This change can cause discomfort to exoskeleton wearers. It is possible to solve these problems by changing the mechanical design, but if not, the control algorithm should be designed considering the change of the gait pattern.

4

An Adaptive Stepsize RRT Planning Algorithm for Open Chain Robots

4.1 Introduction

In this chapter, we propose an adaptive stepsize RRT path planning algorithm for open chain robots in which only a minimum obstacle size parameter is required as input. Exploiting the structure of an open chain's forward kinematics as well as a standard inequality bound on the operator-induced matrix norm, we derive a maximum Cartesian displacement bound between two configurations of the same robot and use this bound to determine a maximum allowable stepsize at each iteration. We also extended this bound to latent space when there is a differentiable mapping from the latent space to the joint space. In addition, we have proposed a relaxed connection condition that can improve the existing RRT performance. Numerical experiments involving a ten-dof planar open chain and a seven-axis industrial robot design demonstrate the practical advantages of our algorithm over

standard fixed-stepsize RRT planning algorithms.

This chapter is organized as follows. The main inequality bound is derived in Section 4.2, while the accompanying adaptive stepsize RRT algorithm is presented in Section 4.3. Section 4.4 presents experimental results using our planning algorithm for a ten-dof planar open chain and a seven-axis industrial robot manipulator. We conclude with some suggestions for extension to more complex chains, and also to planning in spaces of reduced dimension.

4.2 A Cartesian Displacement Bound for Open Chains

We first cover some preliminaries on matrix norms; see, e.g., [39] for a review of standard results on matrix and operator norms. Given two vector norms $\|\cdot\|_X$ on $\mathbb{R}^n$, $\|\cdot\|_Y$ on $\mathbb{R}^m$, and a matrix $A \in \mathbb{R}^{m\times n}$, the matrix operator norm induced from the two vector norms is defined as follows:

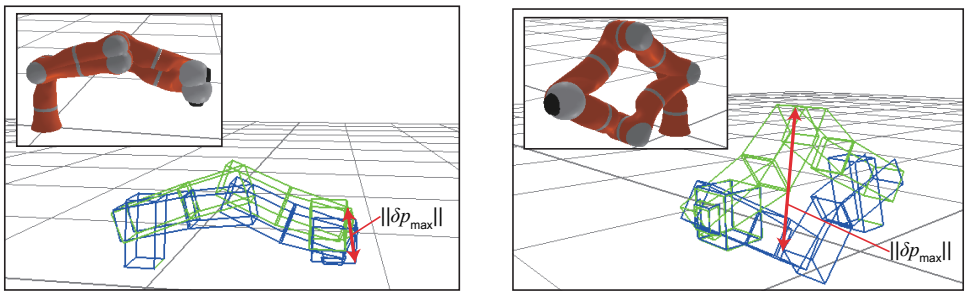
$$\|A\|_{XY} = \sup_{x\neq 0} \frac{\|Ax\|_Y}{\|x\|_X}. \tag{4.2.1}$$

The following inequality then always holds:

$$\|Ax\|_Y \leq \|A\|_{XY} \cdot \|x\|_X$$

For the basic vector norms, e.g., the 1-norm $\|\cdot\|_1$, the Euclidean norm $\|\cdot\|_2$, and the sup-norm $\|\cdot\|_\infty$, the corresponding induced matrix norm applied to $A = (a_{ij}) \in \mathbb{R}^{m\times n}$ are respectively

$$\begin{aligned}
\|A\|_{1,1} &= \max_j \sum_{i=1}^{m} |a_{ij}| \\
\|A\|_{2,2} &= \rho_{\max}(A) = \sqrt{\lambda_{\max}(A^T A)} \\
\|A\|_{\infty,\infty} &= \max_i \sum_{j=1}^{n} |a_{ij}|.
\end{aligned}$$



(a) (b)

Figure 4.1: Two examples of the maximum Cartesian space displacement $\|\delta p_{\max}\|$: (a) $\|\delta p_{\max}\|$ occurs at the end-effector; (b) $\|\delta p_{\max}\|$ occurs at the elbow.

It is well-known that the Frobenius norm $\|\cdot\|_F$, defined as

$$\|A\|_F = \sqrt{\mathrm{Tr}(A^T A)}$$

is not an induced norm of any vector norm. The following inequality also holds for all $A \in \mathbb{R}^{n\times n}$:

$$\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2.$$

We now derive our main result which provides, for an open kinematic chain whose task space is the special Euclidean group SE(3) of rigid body transformations, an inequality bound on the maximum Cartesian position displacement in the task space for a given displacement in the input configuration space.

Proposition 4.1. *Given an m-joint open chain robot with forward kinematics (2.3.47), for a joint displacement $\theta \mapsto \theta + \delta\theta$, the end-effector Cartesian position displacement $p \mapsto p + \delta p$ is approximately bounded by $\|\delta p\|_Y \leq \|A\|_{XY} \cdot \|\delta\theta\|_X$, where*

$$A = \left[\omega_1' \times p + v_1' | \cdots | \omega_m' \times p + v_m'\right] \in \mathbb{R}^{6\times m}, \tag{4.2.2}$$

$\|\cdot\|_X$ denotes a norm in joint configuration space, $\|\cdot\|_Y$ denotes a norm in the Cartesian position space $\mathbb{R}^3$, and $\|\cdot\|_{XY}$ denotes the induced matrix operator norm as defined in (4.2.1).

Proof. Noting that $\dot{p} = \omega_s \times p + v_s$ with ω_s and v_s obtained from (2.3.51)-(2.3.52), $\dot{p}$ can be written explicitly as

$$\dot{p} = \sum_{i=1}^{m} \left([\omega_i']p + v_i'\right)\dot{\theta}_i \tag{4.2.3}$$

$$= \left[[\omega_1']p + v_1' \,|\, \cdots \,|\, [\omega_m']p + v_m'\right]\dot{\theta}, \tag{4.2.4}$$

from which it follows that

$$\delta p \approx \left[\omega_1' \times p + v_1' \,|\, \cdots \,|\, \omega_m' \times p + v_m'\right]\ \delta\theta. \tag{4.2.5}$$

The approximate bound on $\|\delta p\|$ now follows. □

Note that the vector $\omega_i' \times p + v_i' \in \mathbb{R}^3$ represents the contribution of the i-th joint velocity $\dot{\theta}_i$ to the overall end-effector velocity $\dot{p}$.

Now suppose each link i has the shape of a convex polytope, and let $\mathcal{B}_i = \{\mathbf{p}_{i1}, \mathbf{p}_{i2}, \ldots\}$ be the set of link vertices for link i. Let $p_{ij} \in \mathbb{R}^3$ be the coordinates of link vertex $\mathbf{p}_{ij}$ expressed in fixed frame coordinates. Given some joint space displacement $\delta\theta \in \mathbb{R}^m$ from its current configuration θ, there will exist some link vertex $\mathbf{p}_{\max}$ on the robot at which the corresponding Cartesian space displacement will be maximized; denote this maximum displacement by $\delta p_{\max} \in \mathbb{R}^3$ (Fig. 4.1 illustrates the maximum Cartesian space displacement for two different pairs of initial-final configurations). The following corollary then holds.

Corollary 4.2. *The maximum Cartesian space displacement of the m-link open*

chain is approximately bounded by

$$\begin{aligned} \|\delta p_{max}\|_Y &= \max_{i,j} \|\delta p_{ij}\|_Y \\ &\leq \max_{i,j} \|A_{ij}\|_{XY} \|\delta\theta\|_X \end{aligned} \tag{4.2.6}$$

where

$$A_{ij} = \left[\omega_1' \times p_{ij} + v_1' \,|\cdots|\, \omega_i' \times p_{ij} + v_i'\right], \tag{4.2.7}$$

and the search for the maximum occurs over the range $i = 1, \ldots, m$, $p_{ij} \in \mathcal{B}_i$.

The proof follows straightforwardly from a repeated application of the earlier proposition.

A similar result can be derived for input spaces other than the joint space, e.g., a lower-dimensional space, called the latent space, together with a differentiable mapping from the latent space to the joint space. Such situations arise when, e.g., dimension reduction methods are employed to construct a lower-dimensional representation of the configuration space based on a collection of sample trajectories (see, for example, [1]). Let $g : \mathbb{R}^d \to \mathbb{R}^m$, $x \mapsto \theta = g(x)$ be the local coordinate representation of the differentiable mapping from the latent space to the joint configuration space, with $d \leq m$. Let $\delta x \in \mathbb{R}^d$ be a displacement in latent space and $\delta p_{\max}$ be the corresponding maximum Cartesian space displacement for a point on the robot. Denote by $\|\cdot\|_Z$ the norm on $\mathbb{R}^d$. The following corollary then holds.

Corollary 4.3. *The maximum Cartesian space displacement* δp_{max} *is approximately bounded by*

$$\|\delta p_{max}\|_Y \leq \max_{i,j} \|A_{ij}\|_{XY} \left\| \frac{\partial g}{\partial x} \right\|_{ZX} \|\delta x\|_Z. \tag{4.2.8}$$

The corollary follows directly from the fact that $\delta\theta \approx \frac{\partial g}{\partial x}\delta x$.

4.3 Adaptive Stepsize RRT

Based on the inequalities derived in the previous section, we now describe our adaptive stepsize RRT planning algorithm. A maximum Cartesian displacement value Δ in the workspace is first specified; for example, Δ can be set to be the width of the smallest obstacle in the workspace. Setting the right-hand side of Equation (4.2.6) to Δ, i.e.,

$$\|\delta p_{\max}\| \leq \max_{i,j} \|A_{ij}\| \|\delta\theta\| = \Delta, \tag{4.3.9}$$

at each joint space configuration the stepsize is then set to

$$\|\delta\theta\| = \frac{\Delta}{\max_{i,j} \|A_{ij}\|}. \tag{4.3.10}$$

Similarly, for planning in the latent space, the right-hand side of (4.2.8) is set to Δ, and the stepsize is determined as follows:

$$\|\delta p_{\max}\| \leq \max_{i,j} \|A_{ij}\| \left\| \frac{\partial g}{\partial x} \right\| \|\delta x\| = \Delta \tag{4.3.11}$$

$$\|\delta x\| = \frac{\Delta}{\max_{i,j} \|A_{ij}\| \left\| \frac{\partial g}{\partial x} \right\|}. \tag{4.3.12}$$

Note that the stepsizes in (4.3.10) and (4.3.12) depend on the robot's current configuration, and tend to be smaller when a robot's links are fully extended, and larger when the links are folded.

Algorithm 1 describes the main steps of our bounded displacement adaptive stepsize RRT algorithm. Lines 1-7 are the same as for the standard joint space RRT algorithm: sample a random node (θ_{rand}) in the configuration space, find its nearest node (θ_{near}) on the tree and the direction ($\hat{v}_{\text{new}}$) in which to extend the tree. The adaptive stepsize (4.3.10) is evaluated (line 8) and the new node θ_{new} generated in the direction of $\hat{v}_{\text{new}}$ (line 9). If the new node is collision-free and

Algorithm 1 Bounded Displacement Adaptive Stepsize RRT

1: Tree, Δ
2: Tree.AddNode(θ_{init})
3: **While** $i < I_{\max}$ **do**
4: $i \leftarrow i + 1$
5: $\theta_{\text{rand}} \leftarrow$ RandomSampling()
6: $\theta_{\text{near}} \leftarrow$ FindNearestNodeOnTree(Tree, θ_{rand})
7: $\hat{v}_{\text{new}} \leftarrow$ GetNewDirection(θ_{near}, θ_{rand}, Tree)
8: $\|\delta\theta\| \leftarrow$ GetStepsize(θ_{near}, $\hat{v}_{\text{new}}$, Δ)
9: $\theta_{\text{new}} \leftarrow$ Extend(Tree, θ_{near}, $\hat{v}_{\text{new}}$, $\|\delta\theta\|$)
10: **if** CheckConstraints(θ_{new}, constraints) **then**
11: Tree.AddNode(θ_{new})
12: **if** Connect(θ_{new}, θ_{goal}, Δ) **then**
13: [Tree, *flag*] $\leftarrow$ ExtendFurther(Tree, θ_{new}, θ_{goal}, $\hat{v}_{\text{new}}$)
14: **if** *flag* == *true* **then**
15: **return** Path(Tree)
16: **end if**
17: **end if**
18: **end if**
19: **end While**
20: **return** NULL

within the allowed joint range (line 10), then it is added to the tree (line 11). Standard RRT algorithm and the named nodes are illustrated in Figure 4.2.

Lines 12-17 implement the adaptive stepsize planning procedure. *Connect()* (Algorithm 3) determines if the two nodes θ_{new} and θ_{goal} should be connected. In the

Algorithm 2 GetStepsize(θ, $\hat{v}$, Δ)

1: $c \leftarrow \max_{i,j} \|A_{ij}(\theta)\|$
2: $\|\delta\theta\| \leftarrow \frac{\Delta}{c}\|\hat{v}\|$
3: **return** $\|\delta\theta\|$

Algorithm 3 Connect($\theta_{\rm a}$, $\theta_{\rm b}$, Δ)

1: $\mathcal{B}_{\rm a} \leftarrow$ GetEachVertexPosition($\theta_{\rm a}$)
2: $\mathcal{B}_{\rm b} \leftarrow$ GetEachVertexPosition($\theta_{\rm b}$)
3: **if** $\max_{i,j} \|p_{ij}(\theta_{\rm a}) - p_{ij}(\theta_{\rm b})\| \leq \Delta$ **then**
4: **return** *true*
5: **else**
6: **return** *false*
7: **end if**

most basic form of fixed stepsize RRT planning, the connection test is typically done in the joint configuration space, e.g., $\theta_{\rm new}$ and $\theta_{\rm goal}$ are connected if the condition $\|\theta_{\rm new} - \theta_{\rm goal}\| \leq \epsilon$ is satisfied for some arbitrary user-prescribed ϵ. In our case it makes more sense to use $\|\delta\theta\|_{\theta=\theta_{\rm new}}$ in place of ϵ, or noting from Eq. (4.3.10) that $\Delta = \max_{i,j} \|A_{ij}\| \cdot \|\delta\theta\|$, to connect $\theta_{\rm new}$ and $\theta_{\rm goal}$ if

$$\max_{i,j} \|A_{ij}\| \cdot \|\theta_{\rm new} - \theta_{\rm goal}\| \leq \Delta \tag{4.3.13}$$

is satisfied. However, since from Corollary 4.2 we have

$$\begin{aligned} \|\delta p_{\max}\| &= \max_{i,j} \|p_{ij}(\theta_{\rm new}) - p_{ij}(\theta_{\rm goal})\| \\ &\leq \max_{i,j} \|A_{ij}\| \cdot \|\theta_{\rm new} - \theta_{\rm goal}\|, \end{aligned}$$

Connect() uses the less strict connection test $\|\delta p_{\max}\| \leq \Delta$ to speed up planning. In our later experiments we compare the results of using both the joint space and Cartesian space metrics for the connection test.

Algorithm 4 ExtendFurther(Tree, θ, θ_{goal}, $\hat{v}$)

While $j < J_{\max}$ **do**
  $\|\delta\theta\| \leftarrow$ GetStepsize(θ, $\hat{v}$, Δ)
  $\theta' \leftarrow$ Extend(Tree, θ, $\hat{v}$, $\|\delta\theta\|$)
  if CheckConstraints(θ', constraints) **then**
    Tree.AddNode(θ')
    if $(||\theta' - \theta_{\text{goal}}|| \leq \|\delta\theta\|)$ **then**
      return [Tree, *true*]
    else
      $\theta \leftarrow \theta'$
    end if
  else
    return [Tree, *false*]
  end if
  $j \leftarrow j + 1$
end While
return [Tree, *false*]

If θ_{new} and θ_{goal} are close enough to pass the above connection test, then in principle they could be connected and the algorithm terminated. In practice, θ_{new} and θ_{goal} may be quite distant from each other relative to other connected node pairs (we have observed in our later planar robot experiments that these values sometimes exceed more than thirty times typical values of $\|\theta_i - \theta_{i+1}\|$). Although the condition $\|\delta p_{\max}\| \leq \Delta$ ensures that there is no obstacle larger than Δ between $p_{ij}(\theta_{\text{new}})$ and $p_{ij}(\theta_{\text{goal}}), \forall i, j$, depending on the path taken between θ_{new} and θ_{goal}, a collision cannot be ruled out.

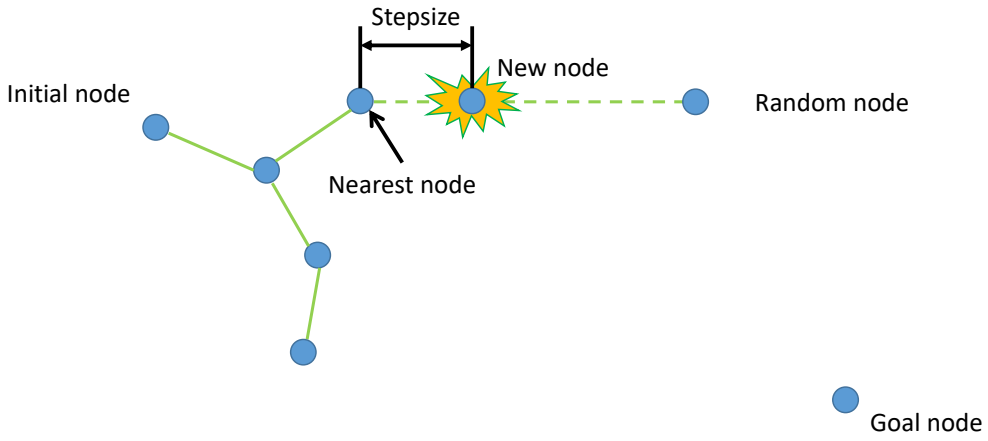


Figure 4.2: The stepsize and the named nodes for RRT.

We therefore invoke the routine *ExtendFurther()* in Algorithm 4, which segments the path between θ_{new} and θ_{goal} into smaller segments of appropriate stepsize, and sequentially checks whether the nodes of these smaller segments are valid. If one of the nodes is not valid, the algorithm returns to the main loop and adds the recently accrued valid nodes to the tree. Specifically, if $\|\delta p_{\max}\| \leq \Delta$, the tree is extended toward θ_{goal} in the direction $\hat{v} = (\theta_{\text{goal}} - \theta_{\text{new}})/\|\theta_{\text{goal}} - \theta_{\text{new}}\|$, with stepsize determined from (4.3.10), while checking for collisions with obstacles and joint limits (lines 2-4). If a collision occurs or joint limits are exceeded at the extended node θ'_{new}, then the algorithm terminates (lines 4 and 12). Otherwise, the algorithm continues to extend the node toward θ_{goal} until the distance $\|\theta'_{\text{new}} - \theta_{\text{goal}}\|$ is less than the stepsize $\|\delta\theta\|_{\theta=\theta'_{\text{new}}}$ evaluated at θ'_{new}, at which point the two nodes are connected.

4.4 Experiments

Our proposed adaptive stepsize RRT algorithm is evaluated through two sets of numerical experiments, the first involving a ten-dof planar open chain, the second involving a standard seven-axis industrial robot design. For the numerical experiments we compare our adaptive stepsize RRT algorithm with two versions of the standard fixed stepsize RRT algorithm and two versions of the RRT-Connect algorithm (in the latter case, the two versions correspond to using a joint space metric versus a Cartesian space metric for the connectivity test). We give a goal bias [40] of probability 5% toward the goal node. Like most RRT-based panning methods, our algorithm is implemented as a bi-directional search algorithm. A non-incremental 3D collision checking algorithm is used for our numerical experiments. The numerical experiments for the ten-dof planar chain and the seven-axis industrial robot arm are performed using code written in C++ and running on an Intel Core i7-4790 3.50 GHz platform with 24 GB RAM.

Although evaluating the operator-induced norm is in general NP-hard, those induced from the one- and two-norms are more straightforward to compute [41, 42]. For our experiments we take the one-norm $\|\cdot\|_1$ in the input configuration space and the two-norm $\|\cdot\|_2$ in the Cartesian task space; the corresponding induced operator norm $\|\cdot\|_{1,2}$ is then evaluated as

$$\|A\|_{1,2} = \max_j \left(\sum_i |a_{ij}|^2 \right)^{1/2}. \tag{4.4.14}$$

Pseudo code for calculating $\|A\|_{1,2}$ is provided in Algorithm 5, where θ denotes the current configuration and $\mathcal{B} = \{\mathcal{B}_1, \cdots, \mathcal{B}_m\}$ is the set of link vertices. See, e.g., [31] for a treatment of how to convert Denavit-Hartenberg parameter-based forward kinematics into product of exponentials form. For comparison purposes we

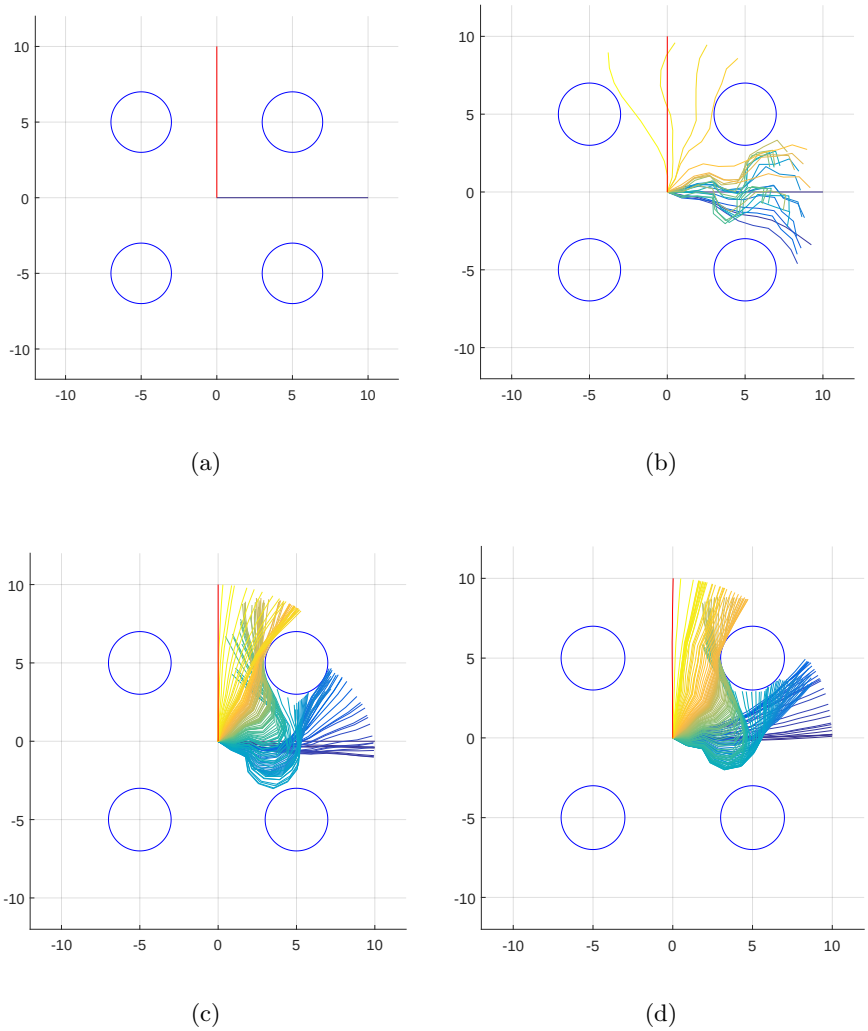


Figure 4.3: Experiments with a ten-dof planar open chain. (a) Circular obstacles are placed symmetrically about the robot base, the initial arm posture is shown in blue, the final posture is shown in red. (b) Results of using standard RRT in joint space with stepsize 0.5; observe that collisions occur with the upper-right obstacle. (c) Results of adaptive stepsize RRT in joint space with $\Delta = 1.5$. (d) Results of adaptive stepsize RRT in latent space with $\Delta = 1.5$.

also perform the same set of experiments using $\|\cdot\|_2$ for the input configuration space. We observe $\|A\|_{2,2}$ is bigger than $\|A\|_{1,2}$, and that all measures, with the exception of elapsed time, vary by a factor of two; the mean and maximum values of $\|\delta p_{\max}\|$ are reduced by half while the number of iterations and path length are doubled. Results of experiments using the two-norm are included in Table 4.1 for comparison purposes.

Algorithm 5 OperatorNorm_12($\theta, \mathcal{B}$)

```
1: a_max ← −inf
2: J_S ← getSpaceJacobian(θ)
3: For i = 1 to m
4:   For j = 1 to numberOfVertexOfLink_i
5:     For k = 1 to i
6:       ret ← ‖ω_k × p_ij + v_k‖_2
7:       if ret > a_max then
8:         a_max ← ret
9:       end if
10:    end For
11:  end For
12: end For
13: return a_max
```

4.4.1 Ten-Dof Planar Robot

We consider a planar ten-dof open chain consisting of ten unit-length links connected by revolute joints. Four circular obstacles of radius 2 are placed symmetrically about the robot base as shown in Fig. 4.3-(a). In the initial configuration

the arm is outstretched along the horizontal axis (indicated in blue); in the goal configuration (indicated in red), the arm is outstretched along the vertical axis.

For our first set of experiments, we fix the stepsize to 0.5 and perform 100 trials of the standard fixed stepsize RRT algorithm in joint space; results are labelled "RRT joint space metric" in Table 4.1 and one result from a single trial run is shown in Fig. 4.3-(b). The mean value of $\|\delta p_{\max}\|$ taken over each step is 1.55 ± 0.95, while the maximum value of $\|\delta p_{\max}\|$ averaged over 100 trials is 4.37 ± 0.75. For the fixed stepsize value 0.5, in the latter case the mean maximum displacement exceeds the diameter of the obstacle, and as can be seen from the results, the planner is unable to detect obstacles, producing infeasible paths that result in collisions. Reducing the stepsize further to 0.3, the standard fixed stepsize RRT algorithm still cannot detect the workspace obstacles and fails to produce feasible collision-free paths, despite a significant increase in the number of iterations and computation times as seen from Table 4.1.

We also pair the standard fixed stepsize joint space RRT algorithm with the Cartesian space metric condition $\|\delta p_{\max}\| \leq \Delta$ for the node connection test: for each link vertex, we calculate its maximum Cartesian space displacement; the nodes are connected if this value is less than the user-specified $\Delta = 1.5$. To find an appropriate fixed stepsize for meaningful comparison, the mean value of $\|\delta p_{\max}\|$ is measured by gradually decreasing the stepsize as shown in Fig 4.4. In this case the stepsize is selected such that the mean value of $\|\delta p_{\max}\|$ is similar to the corresponding value obtained for our adaptive stepsize RRT when $\Delta = 1.5$. Because our user-specified parameter is Δ, which limits $\|\delta p_{\max}\|$, we compare the cases where the mean values of $\|\delta p_{\max}\|$ are similar. The average stepsize of our RRT adaptive stepsize algorithm is also indicated by diamonds. Results of these trials are

also summarized in Table 4.1 (labelled as "RRT Cartesian space metric"). Compared to fixed stepsize RRT using a joint space metric for the connection test, the RRT Cartesian space metric algorithm has significantly better convergence. Even with the stepsize reduced to a fourth of its initial value 0.3, the number of iterations is considerably reduced and the elapsed time is decreased by approximately 95%. The results imply that despite the additional computation imposed by the calculation of Cartesian displacements for each link vertex, performance improves considerably. It is worth emphasizing, however, that in both versions of the fixed stepsize RRT algorithm, it is difficult to know in advance an appropriate stepsize; often the stepsize must be chosen in a cumbersome trial-and-error fashion. As shown in the Fig. 4.5-(a), by trial-and-error we find that when the stepsize is less than 0.1, the maximum value of $\|\delta p_{\max}\|$ becomes smaller than 1.5, resulting in a feasible path. Note that for algorithms using Cartesian connectivity conditions, the maximum value of $\|\delta p_{\max}\|$ converges to the Δ value set a priori. This is a natural outcome of not including the ExtendFurther function in Algorithm 4.

We now perform 100 trials of our adaptive stepsize RRT algorithm, setting $\Delta = 1.5$ and using (4.3.10) to automatically determine the stepsize. As shown in Table 4.1, the mean value of $\|\delta p_{\max}\|$ for our adaptive stepsize RRT is 0.30 ± 0.20. When the stepsize for the RRT Cartesian space metric algorithm is fixed to 0.07, the mean value of $\|\delta p_{\max}\|$ is similar to that of our adaptive stepsize RRT. Comparing these cases, our adaptive stepsize RRT has the smallest number of iterations and also the shortest planning times. (The additional step (Algorithm 4) needed in the adaptive stepsize RRT algorithm is counted as part of the number of iterations). The path length, which is calculated in joint space, is also the shortest when adaptive stepsize RRT is applied.

Our adaptive stepsize RRT algorithm is also applied and compared to RRT-Connect. Similar to the previous experiments, we set $\Delta = 1.5$; the stepsizes for RRT-Connect are then set to 0.07 and 0.075 (so that the mean values of $\|\delta p_{\max}\|$ are similar to those obtained for our adaptive stepsize RRT algorithm). Results are also shown in Table 4.1. Compared to our original adaptive stepsize RRT method, RRT-Connect has fewer iterations and also shorter planning times. Planning times increase slightly when the Cartesian space metric condition is applied to RRT-Connect, but the number of iterations is similar. Path lengths are also found to decrease.

Finally, we combine our adaptive stepsize RRT method with RRT-Connect. The RRT-Connect adaptive stepsize algorithm has the smallest number of iterations, the shortest planning times, and the shortest path lengths. Again, it is worth emphasizing that in standard fixed stepsize RRT algorithms, the stepsize needs to be specified a priori by the user, typically in trial-and-error fashion, while in our adaptive stepsize RRT algorithm, only the very intuitive obstacle minimum size parameter Δ needs to be specified.

Table 4.1: RRT performance statistics for a ten-dof planar robot

	stepsize	Δ	$\|\delta p_{\max}\|$		# iterations	time(s)	path length
			mean	max			
RRT joint space metric	0.5	-	1.55 ± 0.95	4.37 ± 0.75	6090.9 ± 4654.0	0.74 ± 0.79	122.9 ± 32.1
RRT joint space metric	0.3	-	0.97 ± 0.63	3.32 ± 0.55	95767.4 ± 81743.6	52.51 ± 70.44	309.7 ± 79.3
RRT Cartesian space metric	0.07	1.5	0.30 ± 0.20	1.34 ± 0.14	4254.8 ± 1722.1	0.61 ± 0.39	274.5 ± 60.5
RRT Cartesian space metric	0.06	1.5	0.26 ± 0.18	1.35 ± 0.15	5199.7 ± 1840.8	0.82 ± 0.46	323.6 ± 65.5
RRT adaptive stepsize $\|\cdot\|_{1,2}$	-	1.5	0.30 ± 0.20	1.18 ± 0.20	3337.1 ± 1282.0	0.46 ± 0.27	259.4 ± 60.5
RRT adaptive stepsize $\|\cdot\|_{2,2}$	-	1.5	0.15 ± 0.10	0.67 ± 0.07	7552.9 ± 2621.5	2.72 ± 1.21	495.3 ± 110.4
RRT-Connect joint space metric	0.07	-	0.30 ± 0.19	0.94 ± 0.09	3655.9 ± 1127.7	0.27 ± 0.13	242.1 ± 53.3
RRT-Connect joint space metric	0.075	-	0.33 ± 0.21	1.01 ± 0.10	3310.9 ± 983.8	0.23 ± 0.11	227.0 ± 45.7
RRT-Connect Cartesian space metric	0.07	1.5	0.32 ± 0.21	1.38 ± 0.10	3689.4 ± 1141.7	0.29 ± 0.14	229.4 ± 48.4
RRT-Connect adaptive stepsize	-	1.5	0.32 ± 0.23	1.37 ± 0.06	2870.5 ± 833.3	0.23 ± 0.10	206.7 ± 40.7

Table 4.2: RRT performance statistics for a ten-dof planar robot in latent space

reference	algorithm	stepsize	Δ	$\|\delta p_{\max}\|$			# iterations	time(s)
				mean	min	max		
fixed stepsize	fixed stepsize	0.1	-	0.56 ± 0.41	0.08 ± 0.047	1.89 ± 0.59	$4.18E+02 \pm 1.93E+02$	0.85 ± 0.39
	fixed stepsize	0.05	-	0.28 ± 0.21	0.03 ± 0.019	0.98 ± 0.31	$1.00E+03 \pm 4.48E+02$	2.05 ± 0.93
($\|\delta\theta\| = 0.3$)	fixed stepsize	0.03	-	0.17 ± 0.12	0.02 ± 0.007	0.62 ± 0.19	$1.94E+03 \pm 8.55E+02$	4.06 ± 1.83
	adaptive stepsize	-	1.5	0.18 ± 0.13	0.02 ± 0.006	0.68 ± 0.13	$1.25E+03 \pm 4.08E+02$	5.71 ± 1.71
adaptive stepsize	fixed stepsize	0.1	-	0.33 ± 0.23	0.04 ± 0.019	1.28 ± 0.34	$6.49E+02 \pm 2.30E+02$	1.31 ± 0.46
	fixed stepsize	0.05	-	0.17 ± 0.12	0.02 ± 0.006	0.70 ± 0.20	$1.44E+03 \pm 5.30E+02$	2.94 ± 1.08
($\Delta = 1.5$)	fixed stepsize	0.03	-	0.10 ± 0.07	0.01 ± 0.004	0.44 ± 0.11	$2.62E+03 \pm 9.16E+02$	5.47 ± 1.95
	adaptive stepsize	-	1.5	0.19 ± 0.13	0.02 ± 0.007	0.70 ± 0.16	$9.50E+02 \pm 3.30E+02$	5.23 ± 1.72

4.4.2 Ten-Dof Planar Robot Case II: Latent Space RRT

We now consider planning in a lower-dimensional latent space. For the construction of the latent space, we use the method of dimension reduction based on the Gaussian process dynamic model as described in [1] and Section 2.2.3, which we now briefly review. Letting $x \in \mathbb{R}^d$ be the latent space variable, the corresponding joint variable $\theta(x) \in \mathbb{R}^m$ is assumed to have a Gaussian density with mean function $m(x)$ and covariance function $k(x, x')$:

$$\theta(x) \sim \mathcal{GP}(m(x), k(x, x')).$$

In addition, we assume the stochastic Markov dynamics process in the latent space:

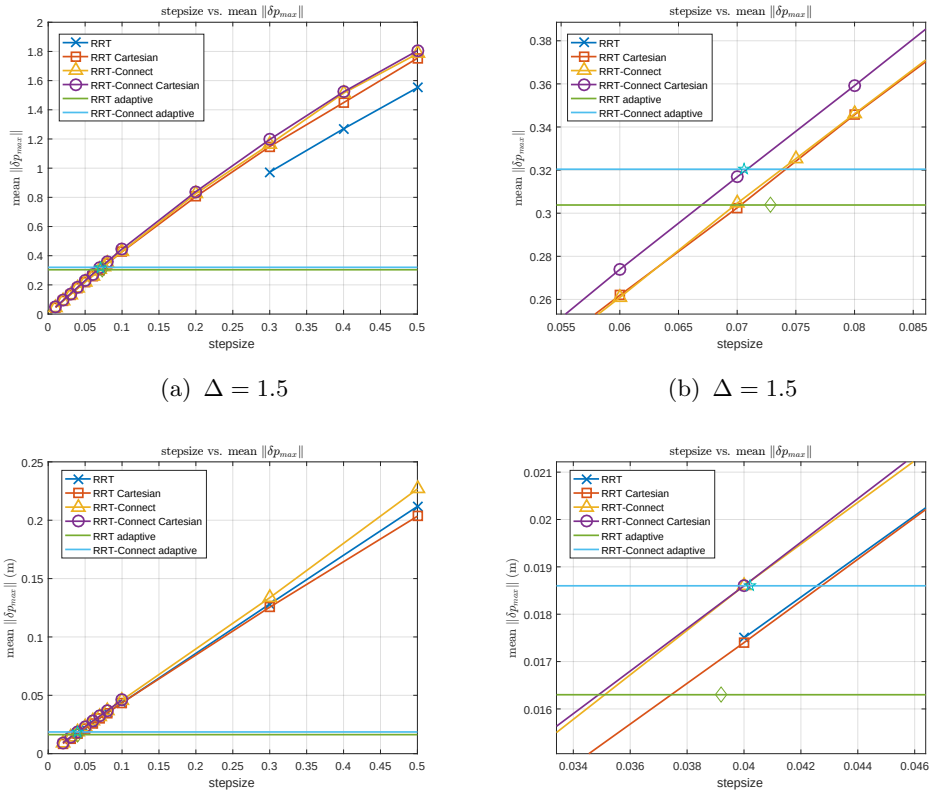
$$x_t \sim \mathcal{GP}(m_X(x_{t-1}), k_X(x_{t-1}, x_{\tau-1})).$$

GPDM is to find latent variables $X = [x_1, \cdots, x_N]^T \in \mathbb{R}^{N\times d}$ for a given representative path in joint space $\Theta = [\theta_1, \cdots, \theta_N]^T \in \mathbb{R}^{N\times m}$. Once the GPDM latent variables $X = [x_1, \cdots, x_N]^T$ are learned from a sample data matrix of representative path $\Theta = [\theta_1, \cdots, \theta_N]^T$, a mapping from new latent space coordinate x^* to joint configuration space coordinate $g(x^*)$ is of the form

$$g(x^*) = \Theta^T K^{-1} k(x^*) \tag{4.4.15}$$

which is the same as Eq. (2.2.37) where $K \in \mathbb{R}^{N\times N}$ is a kernel matrix whose elements are $(K)_{ij} = k(x_i, x_j)$, $k(x) = [k(x_1, x), \cdots, k(x_N, x)]^T \in \mathbb{R}^N$ is a vector of covariances between X and x, and N is the number of nodes of the path. The kernel function $k(x, x')$ is specified by the user; a common choice also used in [1] is the radial basis kernel $k(x, x') = \alpha_1 \exp\left(-\frac{\alpha_2}{2}\|x - x'\|^2\right)$ and we use it. The derivative of $g(x)$ is then

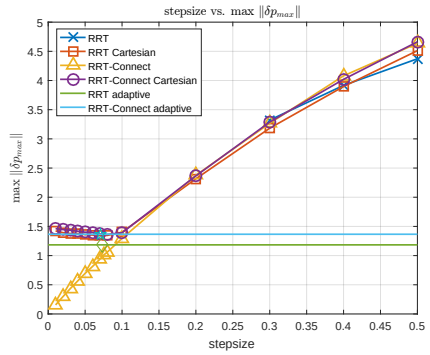
$$\frac{\partial g}{\partial x} = \Theta^T K^{-1} \frac{\partial k(x)}{\partial x}, \tag{4.4.16}$$



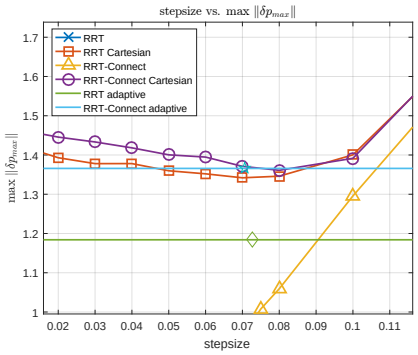
(a) $\Delta = 1.5$ (b) $\Delta = 1.5$

(c) $\Delta = 0.07m$ (d) $\Delta = 0.07m$

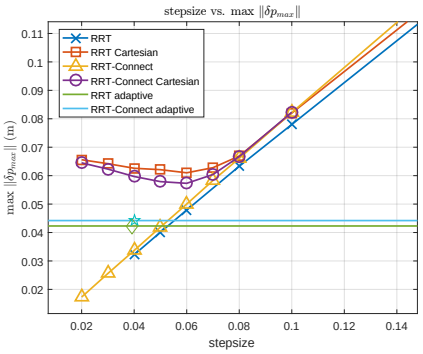
Figure 4.4: Stepsize versus mean $\|\delta p_{\max}\|$. Average stepsizes for RRT with adaptive stepsize and RRT-Connect with adaptive stepsize are marked with diamonds and stars, respectively. (a) 10-dof planar robot. (b) An enlarged view of the region of overlap for the 10-dof planar robot. (c) 7-dof industrial arm. (d) An enlarged view of the region of overlap for the 7-dof industrial arm.



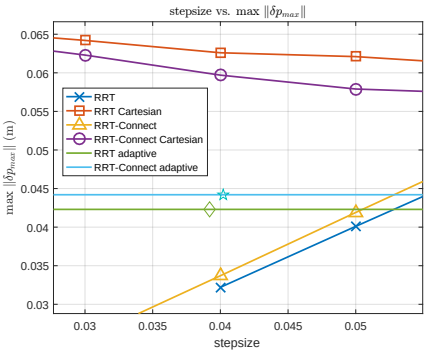
(a) $\Delta = 1.5$



(b) $\Delta = 0.07m$

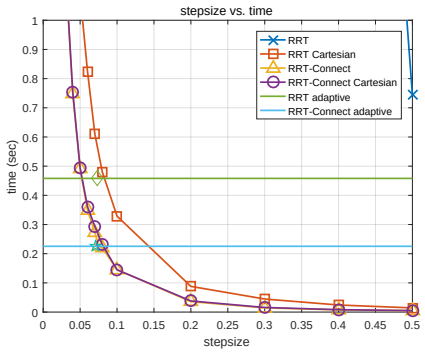


(c) $\Delta = 1.5$

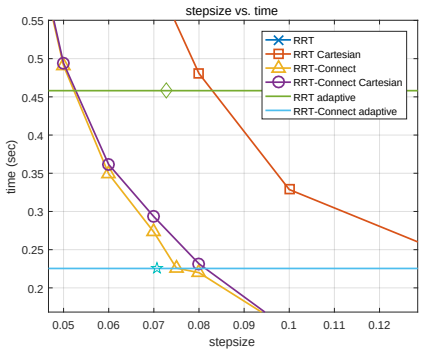


(d) $\Delta = 0.07m$

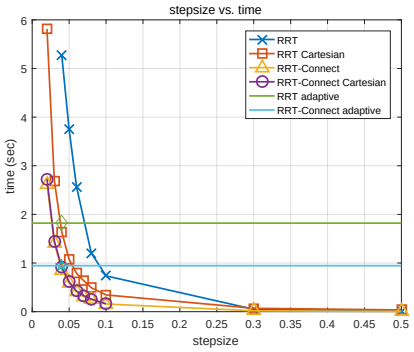
Figure 4.5: Stepsize versus maximum $\|\delta p_{\max}\|$. Average stepsizes for RRT with adaptive stepsize and RRT-Connect with adaptive stepsize are marked with diamonds and stars, respectively. (a) Maximum $\|\delta p_{\max}\|$ for 10-dof planar robot. (b) An enlarged view of the region of overlap for the 10-dof planar robot. (c) Elapsed time for 10-dof planar robot. (d) An enlarged view of the region of overlap for the 7-dof industrial arm.



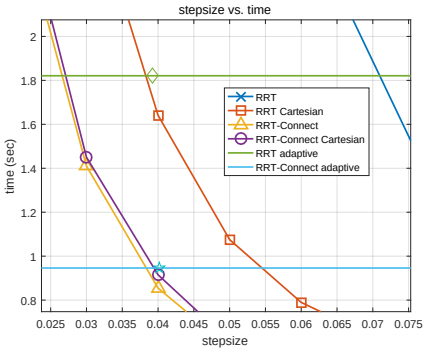
(a) $\Delta = 1.5$



(b) $\Delta = 0.07m$

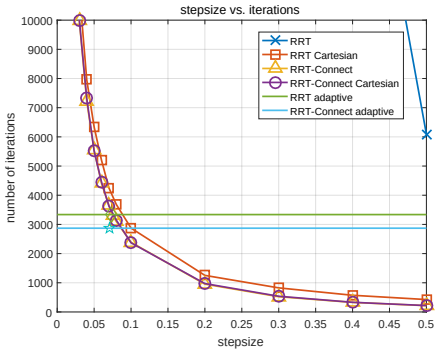


(c) $\Delta = 1.5$

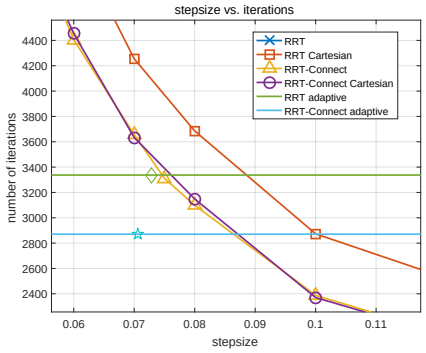


(d) $\Delta = 0.07m$

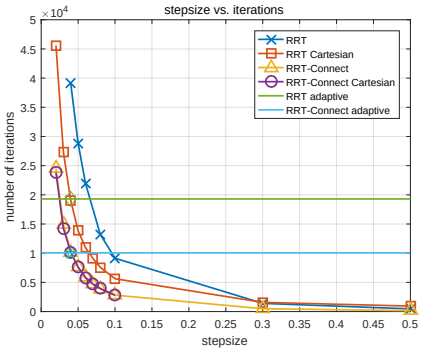
Figure 4.6: Stepsize versus time. Average stepsizes for RRT with adaptive stepsize and RRT-Connect with adaptive stepsize are marked with diamonds and stars, respectively. (a) Elapsed time for 10-dof planar robot. (b) An enlarged view of the region of overlap for the 10-dof planar robot. (c) Elapsed time for 7-dof industrial arm. (d) An enlarged view of the region of overlap for the 7-dof industrial arm.



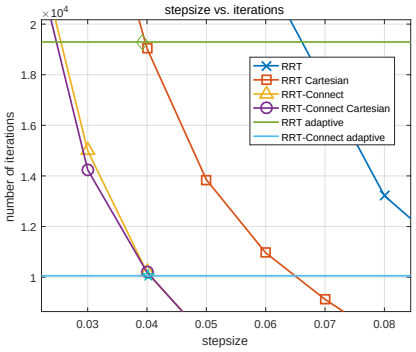
(a) $\Delta = 1.5$



(b) $\Delta = 0.07m$

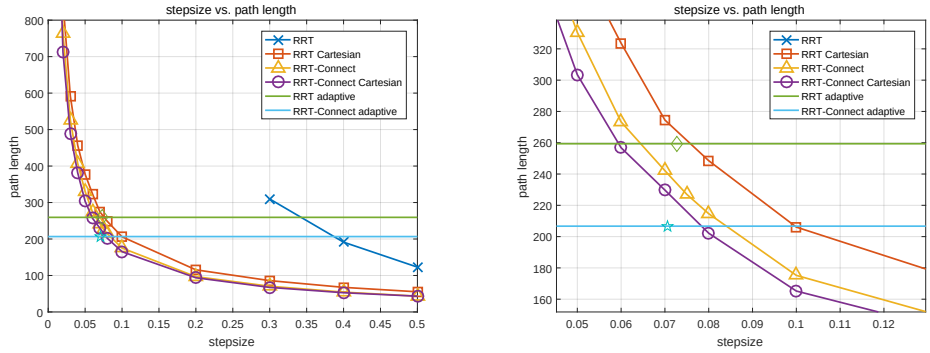


(c) $\Delta = 1.5$

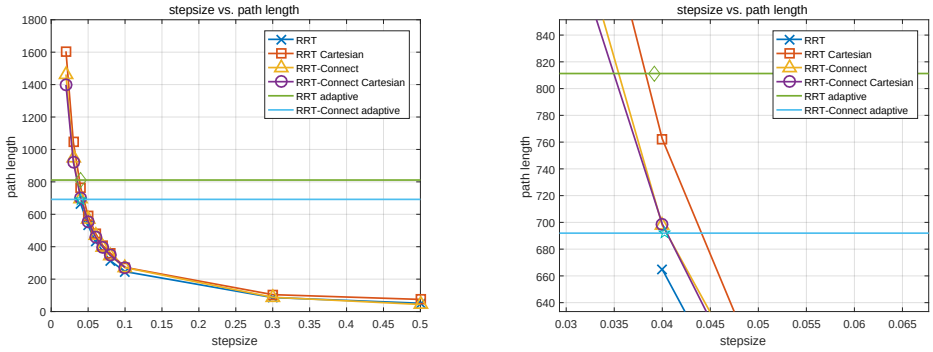


(d) $\Delta = 0.07m$

Figure 4.7: Stepsize versus number of iterations. Average stepsizes for RRT with adaptive stepsize and RRT-Connect with adaptive stepsize are marked with diamonds and stars, respectively. (a) Number of iterations for 10-dof planar robot. (b) An enlarged view of the region of overlap for the 10-dof planar robot. (c) Number of iterations for 7-dof industrial arm. (d) An enlarged view of the region of overlap for the 7-dof industrial arm.



(a) $\Delta = 1.5$ (b) $\Delta = 0.07m$



(c) $\Delta = 1.5$ (d) $\Delta = 0.07m$

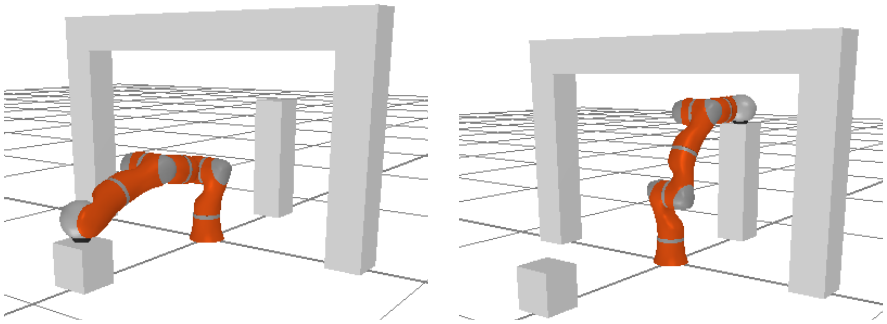
Figure 4.8: Stepsize versus path length. Average stepsizes for RRT with adaptive stepsize and RRT-Connect with adaptive stepsize are marked with diamonds and stars, respectively. (a) Path length for 10-dof planar robot. (b) An enlarged view of the region of overlap for the 10-dof planar robot. (c) Path length for 7-dof industrial arm. (d) An enlarged view of the region of overlap for the 7-dof industrial arm.

where $\frac{\partial k(x)}{\partial x} \in \mathbb{R}^{N \times d}$, and the i-th row of $\frac{\partial k(x)}{\partial x}$ is $\frac{\partial k(x,x_i)}{\partial x}$. Then Corollary 4.3 becomes

$$\|\delta p_{\max}\|_Y \leq \max_{i,j} \|A_{ij}\|_{XY} \left\| \Theta^T K^{-1} \frac{\partial k(x)}{\partial x} \right\|_{ZX} \|\delta x\|_Z. \tag{4.4.17}$$

Corollary 4.3 is now applied for the mapping from the latent space to the joint configuration space as defined above. Ten paths $(\Theta_1, \cdots, \Theta_{10})$ generated by the adaptive stepsize RRT algorithm in the previous section ($\Delta = 1.5$) are randomly chosen, and the corresponding three-dimensional latent spaces are constructed. For comparison, ten paths generated by the basic fixed stepsize RRT algorithm in the previous section (stepsize = 0.3) are randomly chosen, and the corresponding three-dimensional latent spaces are constructed. For each latent space, we perform adaptive stepsize RRT planning ten times with Δ set to 1.5 and the stepsize automatically determined from Equation (4.3.12). Since the latent space dimension is three, we replace Algorithm 3 and Algorithm 4 by the usual connection test used in standard fixed stepsize RRT planning (but now used in the latent space). In addition, we perform basic fixed stepsize RRT ten times with decreasing stepsize $\|\delta\theta\|$ from 0.1 to 0.03. Results of our numerical experiments are summarized in Table 4.2.

As shown in Table 4.2, when basic RRT is applied to GPDM latent space for basic RRT path, we can see that the maximum of $\|\delta p_{\max}\|$ exceeds Δ, generating an infeasible path depending on the stepsize. On the other hand, if adaptive stepsize RRT is applied, a feasible path is generated without a trial-and-error process. In the GPDM latent space for adaptive stepsize RRT path, if stepsize is 0.1, it may be an infeasible path, but it generates path more stable than basic RRT's GPDM latent space. Therefore, it can be seen that planning with basic RRT in latent space is affected not only by stepsize but also by learning objects. On the other



(a) (b)

Figure 4.9: Experiments with a seven-axis industrial robot arm: (a) Initial configuration; (b) Goal configuration.

hand, when adaptive stepsize RRT is used, it can be seen that a constant level of the path is generated irrespective of the quality of the reference path. (Compare the $\|\delta p_{\max}\|$ values of the two cases.) On the other hand, the Cartesian space metric condition is not applied to this experiment, so the adaptive stepsize RRT algorithm is slower. Nevertheless, we want to emphasize the advantage of not having to determine the stepsize as a trial-and-error fashion to get the feasible path.

4.4.3 Seven-DoF Industrial Robot Arm

We now evaluate the performance of our adaptive stepsize algorithm for a seven-axis industrial robot manipulator of the type shown in Fig. 4.9. For easier calculation of the positions of the link vertices, each link is modeled as a three-dimensional rectangular box. The objective is to find a collision-free path from the initial (left figure) to the final configuration (right figure). The robot's maximum length is set to 1 m, and for this problem we set $\Delta = 7cm$ for the variable stepsize algorithm. For the standard RRT algorithm, we set the fixed stepsize initially to

0.1 and successively reduce its value to 0.02 (memory overflow occurs due to the high number of nodes generated, thereby preventing any further reduction of the stepsize). The results of our numerical experiments are summarized in Table 4.3.

For the standard fixed stepsize RRT algorithm, we first compare the use of the joint space metric versus the Cartesian space metric for the stepsize value 0.1. Using the Cartesian space metric leads to a nearly two-fold improvement in efficiency. However, note that for this case the maximum value of $\|\delta p_{\max}\|$ (8.14 ± 0.51) is greater than Δ (7.0). What this implies is that collisions with small objects may potentially occur that could go undetected. This example highlights one of the pitfalls of using the Cartesian space metric without using the accompanying inequality bound: the range of $\|\delta p_{\max}\|$ cannot be accurately estimated since the bound is not known. Furthermore, when the stepsize is 0.3 and 0.5, it fails to satisfy the Cartesian condition and oscillates near the goal, failing to connect the two trees. After repeated numerical trials, we determine that a stepsize value of 0.0375 leads to a mean value $\|\delta p_{\max}\|$ of 1.63 ± 0.65, which is approximately equal to the mean $\|\delta p_{\max}\|$ when Δ of adaptive stepsize RRT is 7. The mean values of $\|\delta p_{\max}\|$ for the changes in stepsize are shown in Fig. 4.4. As shown in Fig. 4.5-(c), when the stepsize becomes smaller than 0.085, the maximum value of $\|\delta p_{\max}\|$ becomes smaller than $\Delta = 7\,cm$, and the resulting path becomes feasible.

The computational performance of our adaptive stepsize RRT algorithm is approximately of the same order as the fixed stepsize RRT algorithm with stepsize values in the range 0.035-0.04. Compared to when the fixed stepsize is set to 0.0375, the number of iterations for the adaptive stepsize RRT algorithm is reduced by about 7%, while the planning time is reduced by about 5%. Note, however, that the maximum displacement for the adaptive stepsize case (4.20 ± 0.32) is smaller than the the corresponding maximum displacements for the fixed stepsize

case (6.28 ± 0.66).

A performance comparison of RRT-Connect versus our adaptive stepsize algorithm is shown in Table 4.3. With $\Delta = 7$, all of the performance indices, i.e., the number of iterations, planning time, and path length, between the adaptive stepsize RRT-Connect algorithm and RRT-Connect are very similar. The results suggest that the advantages of the adaptive stepsize RRT algorithm appear to amplify with increasing degrees of freedom.

In the connectivity test, two adjacent points $\theta_1, \theta_2 \in \mathbb{R}^d$ in joint configuration space must be within a $d-$dimensional sphere of radius $\frac{\Delta}{\max_{i,j} \|A_{i,j}\|}$. When applying the Cartesian connectivity condition, two adjacent points $p_1(\theta_1), p_2(\theta_2) \in \mathbb{R}^3$ must lie in the three-dimensional sphere of radius Δ. As a result, despite the increased computational complexity, our analysis would suggest that planning times are faster when using the Cartesian condition for $d >> 3$.

Finally, we compare RRT-connect with classic RRT adaptive stepsize. The classic RRT adaptive stepsize algorithm continues to extend toward the target node when our Cartesian connectivity test condition is satisfied. (In our algorithm, we execute the ExtendFurther() function when the Connect function is satisfied.) RRT-Connect, on the other hand, believes that the nearest node can be connected to the target node from the beginning, so it continues to extend to the target node as long as there is no collision between them. The target node acts like a sink in a potential field, drawing in the nearest node. RRT-connect is equivalent to this sink affecting the entire configuration space, whereas classic RRT with adaptive stepsize affects only nodes within a certain radius that satisfies the following equation:

$$\max_{i,j} \|A_{ij}\| \cdot \|\theta_{new} - \theta_{goal}\| \leq \Delta. \tag{4.4.18}$$

As a result, it can be seen that RRT-connect, which affects the entire configuration space, does find the path more quickly. Due to these factors, the Cartesian space metric condition applied to the RRT-Connect may degrade performance.

On the other hand, when combining RRT-connect with adaptive stepsize, the resulting algorithm determines in a flexible way whether the stepsize should increase or decrease at the current configuration. As a result, it can be seen that RRT-connect and RRT-Connect adaptive stepsize do not differ greatly in terms of speed. However, if we combine RRT-Connect with the adaptive stepsize algorithm, the above connection check Eq. (4.4.18) seems unnecessary as described above in relation to the potential field interpretation (Connect() and ExtendFurther() functions), since the algorithm will extend the tree to the target node until $\|\theta_{new} - \theta_{goal}\| \leq \epsilon$ is satisfied. The results presented in these tables are a result of including both functions. If these two functions are omitted, it is likely that the performance of RRT-connect with adaptive stepsize can be further improved.

Table 4.3: RRT Statistics for Seven-Axis Robot Arm

	stepsize	Δ	$\|\delta p_{\max}\|$ (cm)		# iterations	time(s)	path length
			mean	max			
RRT joint space metric	0.1	-	4.36 ± 1.56	7.82 ± 0.52	9500.7 ± 8077.6	0.72 ± 1.08	254.7 ± 88.0
RRT Cartesian space metric	0.1	7	4.35 ± 1.61	8.14 ± 0.51	5540.4 ± 2408.8	0.33 ± 0.21	273.9 ± 93.6
RRT Cartesian space metric	0.0375	7	1.63 ± 0.65	6.28 ± 0.66	20978.5 ± 6839.3	1.84 ± 0.86	816.1 ± 255.2
RRT adaptive stepsize	-	7	1.63 ± 0.69	4.20 ± 0.32	19450.5 ± 6277.1	1.74 ± 0.82	814.0 ± 258.6
RRT-Connect joint space metric	0.04	-	1.85 ± 0.86	3.94 ± 0.18	10169.7 ± 3047.0	0.86 ± 0.33	706.4 ± 149.4
RRT-Connect Cartesian space metric	0.04	7	1.86 ± 0.84	4.54 ± 0.33	10120.6 ± 3038.7	0.85 ± 0.33	703.3 ± 148.7
RRT-Connect adaptive stepsize	-	7	1.84 ± 0.96	4.42 ± 0.21	10283.0 ± 3076.1	0.91 ± 0.35	707.1 ± 146.4

4.5 Conclusion

Using a standard operator norm inequality and exploiting the structure of an open chain's forward kinematics equations as captured by the product of exponentials formula [17], we have presented an adaptive stepsize RRT planning algorithm for open kinematic chains. The key result is an approximate bound on the Cartesian displacement of the open chain tip for a given joint space displacement. This bound, which can be computed in real-time and is useful in other motion planning contexts described below, is used to compute, for a given robot configuration, approximate bounds on the maximal deviations of the points on each of the robot's links (each link is modeled as a convex polytope). Not having to manually set the stepsize parameter through a time-consuming trial-and-error process is one of the important advantages of our algorithm.

One by-product of our approach is a Cartesian space metric that can be used to determine whether or not to connect two nodes in standard fixed-stepsize RRT planning algorithms. Despite the increased computation, in many cases the overall performance of the algorithm is enhanced compared to the standard joint space metric threshold test that is more commonly used. The main drawback of having to manually determine an appropriate stepsize still remains, however.

Our adaptive stepsize algorithm has been validated through extensive numerical experiments with a ten-dof planar open chain and a seven-axis industrial arm manipulator. The computation times are generally faster than fixed-stepsize RRT algorithms, and the advantages tend to become more pronounced for robots with higher degrees of freedom. The fact that the stepsizes can be automatically determined simply by the user specifying the size Δ of the smallest workspace obstacles

is the most obvious advantage of our adaptive stepsize algorithm. Our experimental studies further indicate that in many cases the actual maximum displacements are less than the specified Δ, confirming the conservative nature of the displacement bound used.

Recently, for robots with many degrees of freedom, machine learning-based dimension reduction methods, in which a lower-dimensional representation of the configuration space (the latent space) is constructed from a collection of representative motions, have become a popular means of reducing the complexity of the planning problem. Provided the mapping from the latent space to the configuration space is well-characterized and smooth, our method can also be extended to this situation as well.

The bounds used in our algorithm can also be used in other motion planning contexts. Regions in the latent space corresponding to collisions may need to be mapped out, usually by sampling methods; our bound can be used to determine an appropriate sampling resolution from knowledge of the sizes of the obstacles relative to the robot. Although our results have only been established for open chains, robots with more complex structures like closed loops and non-convex link shapes can likely be modeled with appropriate extensions.

5

A Gaussian Process Dynamical Model-Based Planning Method

5.1 Introduction

As robot technology becomes more and more closely related to real life, robots are required to behave like human beings. Planning such motion is a very difficult problem since such movements are very difficult to express mathematically and robots have many joints. To solve these problems, learning from demonstration (LfD) has been attracting attention to robotics because learning from demonstration allows users to define tasks simply. Unlike traditional programming, the end user does not have to design all the controllers to handle all situations. LfD simply allows the end user to program the robot by showing how the human beings perform the given task. Nonetheless, LfD still suffers from the curse of dimensionality arising from many joints of the robot. To solve this problem, many dimensionality reduction methods have been studied. In particular, in the case of human motion,

because each joint move in harmony, one can find a suitable low-dimensional latent space. Then, a given motion and its variation can be expressed very efficiently. In addition, if constraints such as obstacles or joint limits can be well mapped into low dimensional space, conventional sampling-based path planning algorithms can be applied to low-dimensional space.

The existing learning from demonstration methods have devised methods based on the Hidden Markov Model (HMM). However, due to the complexity of the model, HMMs typically require large amounts of data. HMM-based methods have been applied to simple tasks such as walking, sitting and standing [18], pouring water into a cup [19], hand gestures [20], punching and kicking [21], and lifting objects [22]. In the above studies, joint angle trajectories, end effector's three-dimensional Cartesian coordinate system trajectories (i.e., trajectories of the palm) or paths of markers attached to the human body in continuous time domain have been used as learning data. However, the main problem with HMM-based methods is the existence of unnatural discontinuities in describing the data and insufficient the number of data compared to the complexity of the model.

A more recent approach uses Gaussian Mixture Regression (GMR) and Gaussian Mixture Models (GMM) [43, 44]. These algorithms provide a more flexible model for generalizing various tasks. However, these algorithms undergo an optimization process called Expectation Maximization (EM). In this EM process, many parameters fall into the local maxima frequently, which makes the optimization process difficult.

In similar motivations, the Gaussian Process Latent Variable Model (GPLVM) has been studied to deal with high-dimensional human motion [45]. Previous studies using GPLVM have shown that even with a small demonstration, the shared latent space between human postures and their corresponding robot postures[46],

or between human postures and their corresponding animation character postures [29] can be found. The Gaussian Process Dynamical Model (GPDM) [1] has been extended from GPLVM by combining a model that can reflect dynamic characteristics in latent space. GPDM has been applied to visual tracking [36], generating various style gait patterns [47], and optimization using latent space [48].

In this study, we use GPDM to efficiently represent common features of human motion and robot motion. By using GPDM, it was possible to reduce the number of dimensions in a nonlinear way, and it was possible to express efficiently in a low dimensional space with only a small demonstration. In this chapter, we propose a motion planning method that can produce a collision-free path in latent space for reproducibility. This can be achieved by separating the latent space where the collision occurs and the space that does not occur in physical space. We first sampled the latent space at uniform intervals and then labeled these samples as for whether a collision occurs in physical space. Then typical SVM is trained with these labeled samples. Since the mapping between the latent space and the physical space is non-linear, uniform samples in the latent space have uneven distances in the physical space. Therefore, the sampling distance should be selected in consideration of this non-uniformity. We theoretically derive the maximum distance in the physical space of neighboring points in latent space. The relationship between the distance of neighboring points in latent space and the distance of these points in physical space is derived from the forward kinematics expressed in the form of the product of exponentials (POE). After bringing the obstacles into the latent space, a collision-free planner was introduced to generate a collision-free path in the latent space. Using the rapidly-exploiting random tree (RRT) to generate a collision-free path would be the most reasonable choice. However, using the basic RRT in the latent space does not reflect the information about the demonstration

movement. To use information about the demonstration, we use a kernel function derived from GPDM learning to define a potential function, which has the lowest cost along the path of demonstrated motion in latent space. We used a vector field RRT (VF-RRT) that finds a path with a low potential in the potential field, thereby avoiding obstacles while generating motion similar to the learned motion.

The remainder of this chapter is organized as follows. Section 5.2 introduces our learning from demonstration framework, which deals with bringing new poses in joint space into latent space and generating collision-free paths in latent space. Section 5.3 describes the simulation results and the results applied to the robot platform Mahru. The conclusion remark and future directions are described in Section 5.4.

5.2 Learning from Demonstration Framework

In this section, we are going to introduce a learning from demonstration framework using GPDM latent space. The framework ranges from learning demonstrations using GPDM to generating collision-free paths and is shown in Fig. 5.1.

There are two ways to use GPDM latent space for learning from demonstration. The first is when a human demonstration motion and its corresponding robot motion are both given. The second is when the robot's motion is given in the same way as kinesthetic teaching. The difference between the former and the latter is that the shared latent space can be composed of human demonstration and robot motion. Then, using this shared latent space, a new human motion can be transferred to the robot's motion. However, since the collision-free motion is not guaranteed for this transferred motion, the user must generate a new motion that is suitable for the new environment and that is suitable for the new purpose, to

avoid obstacles. Therefore, it is recommended that finding the coordinates in the latent space for the starting point and the goal point corresponding to robot's configuration space and applying our motion planning method.

In order to regenerate a new motion path, two processes must be preceded. First, we have to find a new configuration in the latent space and bring the constraints into the latent space. The former can be achieved by using Gaussian process regression (GPR) and the latter by using a support vector machine (SVM). In order to use SVM, it is necessary to uniformly sample in latent space and verify that the samples meet constraints in workspace or configuration space. Due to nonlinear mapping from latent space to workspace or joint space, even though the samples are uniformly sampled in the latent space, its coordinates become very uneven in the corresponding workspace or joint space. Therefore, it is necessary to predict the maximum deviation distance in the target space. Fortunately, the mapping from latent space to joint space in GPDM is in closed form. Using this, we propose a proposition about the sampling distance in the latent space and the corresponding maximum deviation distance in the workspace.

Finally, in this section, we proposed a motion planning method to reach the target point while avoiding obstacles in the latent space. Since the motion planning is performed in the low-dimensional latent space, the advantages are that it is more efficient than the motion planning in the original high dimensional joint space and that a motion similar to the learned motion can be generated. However, when bringing obstacles from the workspace into the latent space, the RRT-based motion planning method is suitable because the shape of obstacles in latent space is generally very complicated.

The remainder of this section consists of a description of each block in the Figure 5.1. The contents of each block are; how to get a new posture from the

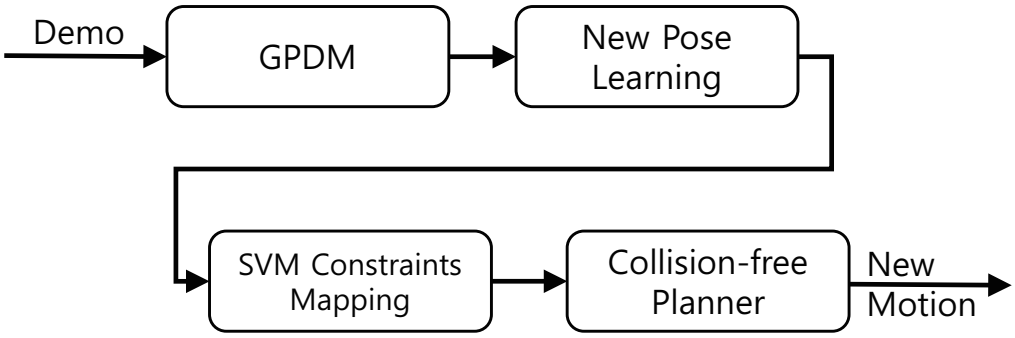


Figure 5.1: Block diagram of learning from demonstration framework using GPDM

workspace into the GPDM latent space, how to bring obstacles from the workspace into the latent space, and VF-RRT, a collision-free planner that generates more natural motion in latent space than other RRT-based motion planning methods.

5.2.1 Learning a New Pose in the Latent Space

Generally, to generate a new motion, the posture of the starting point and the goal point are given in the workspace. Latent space coordinates for these two postures given in the workspace must be found through the joint space coordinates of the robot. This section explains how to find latent space coordinates corresponding to given configuration in joint space.

Let P^* be a new posture given in workspace. Let Z^* be the new configuration in the robot joint space obtained by solving inverse kinematics for P^*. The method of finding the latent space coordinate for the GPDM can be expressed as follows.

$$X^* = \arg\max_x p(Z^*|x, \Gamma_Z), \tag{5.2.1}$$

where Z^* is a new robot pose, X^* is a corresponding latent space coordinate, and

$\Gamma_Z = \{X, Z, \bar{\beta}\}$ is the learned GPDM model. The new coordinate X^* in the latent space can be obtained by minimizing the negative logarithm of the Equation (5.2.1). However, solving the Equation (5.2.1) is very computationally expensive and difficult due to the many local minimums. Using GPR to find the new pose in latent space for new robot pose is a great help in calculation speed[46]. The Gaussian process for given observations Z and its corresponding latent variables X can be expressed as

$$X \sim G.P(0, K(Z, Z)), \tag{5.2.2}$$

where K is a kernel matrix and a kernel function is defined as

$$k(z, z') = \theta_1 \exp\left(-\frac{\theta_2}{2}||z - z'||^2\right) + \theta_3^{-1}\delta_{z,z'}, \tag{5.2.3}$$

where $\bar{\theta} = \{\theta_1, \theta_2, \theta_3\}$ is the hyper parameter. Then the new latent space coordinate X^* for the new observation Z^* is given by

$$X^* = K(Z^*, Z)K^{-1}(Z, Z)X. \tag{5.2.4}$$

5.2.2 Constraints in Latent Space

In joint space, considering constraints (e.g., obstacles) defined in a workspace is very difficult due to the non-linearity of the mapping between these two spaces. Therefore, it is very rare that these constraints are defined as closed-form in joint space. Similarly, it is difficult to consider the constraints defined in joint space in latent space. It is also difficult to bring obstacles in the workspace into the latent space naturally.

However, it is very simple to determine whether any coordinates in the latent space satisfy constraints in joint space or work space. Since the mapping from the

GPDM latent space to the joint space exists in a closed-form, constraint conditions in the joint space can be confirmed by putting latent space coordinates into this closed-form mapping function. Constraints in the workspace can be checked whether the constraints are satisfied by solving inverse kinematics.

Although it is simple to check constraints on arbitrary coordinates in the latent space, there is still a difficulty in optimizing the motion trajectory in the latent space or generating a new motion [49]. We introduce a support vector machine (SVM) which is a machine learning algorithm for motion planning or optimization in latent space while considering constraints defined in joint space or work space. If SVM can be used to directly consider constraints in latent space (that is, without mapping from latent space to workspace or joint space), then the burden of RRT-based motion planning algorithm to check each constraint in physical space per every node in latent space is reduced, so that it can plan a path more efficiently.

SVM is a machine learning method that is used mainly in classification and regression analysis. Here, we apply SVM to our algorithm as a method of using binary classification of whether or not constraints are satisfied. By learning SVM for c constraints, inequality constraints $C_i(X) \leq 0, \quad i = 1, \cdots, c$ can be obtained and we can define the admissible region satisfying the constraint as follows

$$T_c(x) = \{x \in \mathbb{R}^d \mid C_i(x) \leq 0, \text{ for } i = 1, \cdots, c\}. \tag{5.2.5}$$

5.2.3 Mapping Obstacle into Latent Space

Finding the region of obstacles in a joint space is very difficult due to the high dimensionality of joint space as well as the nonlinearity of the map between the joint space and the workspace. If the dimensionality of the joint space is reduced to

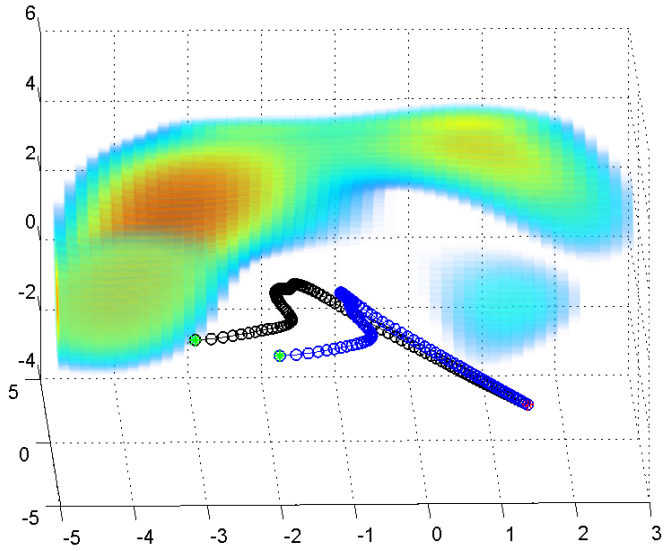


Figure 5.2: Cloud-like objects are the mapped constraints by SVM on the 3-dimensional latent space.

less than three, it is possible to bring obstacles into low-dimensional space. However, if the dimension is reduced by a nonlinear method such as GPDM uniform samples in the latent space become highly uneven samples in the workspace and joint space. To avoid missing important obstacles in the workspace due to this non-uniformity it is very important to predict how much the uniform samples in the latent space will deviate in the workspace. In this section using the previous chapter's Corollary 4.3

$$\|\delta p_{\max}\|_Y \leq \max_{i,j} \|A_{ij}\|_{XY} \left\| \frac{\partial g}{\partial x} \right\|_{ZX} \|\delta x\|_Z.$$

we predicted how much distance the two adjacent samples in the latent space can have in the workspace and proposed a method to determine the appropriate sampling distance.

We consider a planar ten-dof open chain consisting of ten unit-length links connected by revolute joints. Four circular obstacles of radius 2 are placed symmetrically about the robot base as shown in Figure 4.3-(a). In the initial configuration the arm is outstretched along the horizontal axis (indicated in blue); in the goal configuration (indicated in red), the arm is outstretched along the vertical axis. Then ten two-dimensional GPDM latent spaces are learned from ten paths generated by adaptive stepsize RRT.

Since sampling distance $\|\delta x\|_1$ depends on the current latent space coordinate, it is realistic to obtain the average sampling distance in the region of interest and perform uniform sampling. To estimate the average sampling distance $\|\delta x\|_1$, we uniformly divided the region of interest by $N_i = 100$ for each dimension, then

Table 5.1: Example of Sampling Distance and Number of Sampling

$\bar{c}$	Sampling Distance	# of sampling
17.85 ± 6.22	0.0065 ± 0.0028	472656

calculated average sampling distance as following:

$$
\begin{aligned}
c(x) &= \max_{i,j} \|A_{ij}\| \left\| \frac{\partial g}{\partial x} \right\| && (5.2.6)\\
\bar{c}(x) &= \frac{1}{V} \int c(x) dV \\
&\simeq \frac{1}{V} \sum c(x) dV \\
&= \frac{dV}{V} \sum c(x) \\
&= \frac{1}{\prod_i N_i} \sum c(x) && (5.2.7)\\
||\delta x||_1 &= \frac{\Delta}{\bar{c}(x)}. && (5.2.8)
\end{aligned}
$$

For a region of interest in latent space with $([-2.2642, 2.1871] \times [-2.3666, 2.0943])$ and $\Delta = 0.1$, average sampling distance is 0.0065 and required number of sampling is 472656 (see Table 5.1). Actually, dividing the region of interest with such large required number of sampling is computationally inefficient. In case of collision geometry is more complex than this example or dimensionality of latent space is bigger, checking whether all the samples are in collision or not is intractable. To make it computationally tractable and to estimate whether the coordinates of the unsampled area collides in the workspace, we introduce SVM and rely on its regression.

For above $\prod_i N_i$ samples, labels, i.e. collision space or collision-free space, are

assigned and then SVM is trained with RBF kernel

$$k(x, x') = \exp\left(-\frac{1}{\sigma^2}(x - x')^T(x - x')\right). \tag{5.2.9}$$

Depending on the value of σ, decision boundaries are affected[50]. If σ is large, classifier becomes robust to overfitting but does not capture the complexity of data. If σ is small, classifier is prone to overfitting but does capture the complexity of data. In order to classify the obstacles correctly, SVM should capture the complexity of data, i.e., obstacle shape in latent space. So, σ should be small for our problem. After SVM learning, we obtain decision boundary

$$B(x) = \sum_i \alpha_i k(s_i, x) + b \tag{5.2.10}$$

$$\begin{cases} <= 0, & \text{for obstacles,} \\ > 0, & \text{for collision-free space.} \end{cases} \tag{5.2.11}$$

where α_i are coefficients from Lagrange multiplier of optimization procedure of SVM learning, s_i are support vectors, and b is a bias.

Let $\mathcal{D}_{train}$ be a set of samples which are sampled uniformly with $N_i = 100$ along each dimension, and let $\mathcal{D}_{test}$ be a set of samples which are sampled at uniform interval (5.2.8) along each dimension. For the set of kernel width $\sigma = [1, 0.5, 0.1]$ and $\Delta = 0.1$, we measure that: error rate e_{learn} of SVM learning, which is rate of samples having different label by decision boundary function $B(x)$ among $\mathcal{D}_{train}$ samples; generalization error rate e_{cv} obtained from 10-fold cross validation; and misclassification rate e_{test} that is rate of misclassified samples by decision boundary function $B(x)$ from $\mathcal{D}_{test}$. For those misclassified samples of set $\mathcal{D}_{test}$, we measure the distance $dist_{latent}$ from the decision boundary $B(x) = 0$ in latent space, and also measure the distance $dist_{feature}$ in feature space. For ten latent spaces of GPDM, we measure the statistics and their results are summarized

in Table 5.2.

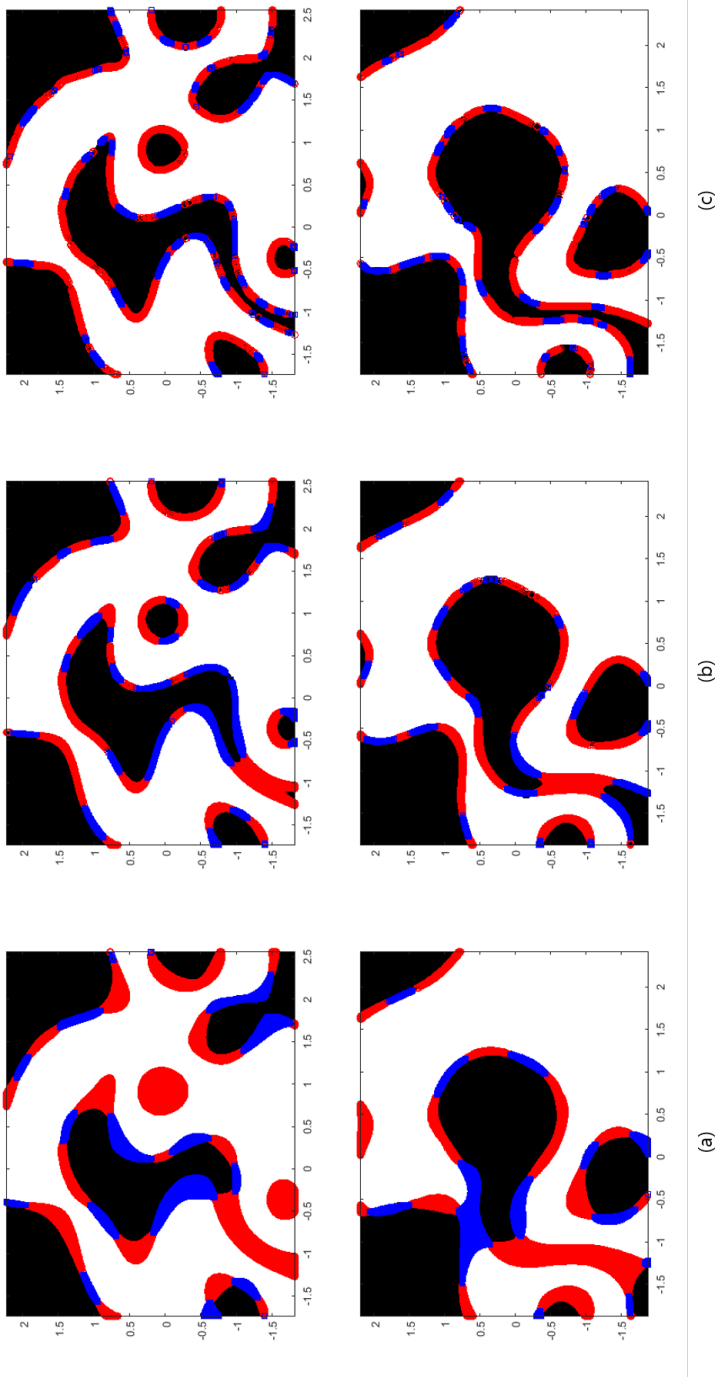


Figure 5.3: Obstacle region (black) and misclassified samples (blue and red). (a) $\sigma = 1$ (b) $\sigma = 0.5$ (c) $\sigma = 0.1$

Figure 5.3 shows obstacle region (black objects) according to the RBF kernel width σ. As mentioned before, small (narrow) obstacle region, which is treated as a just outlier (which is misclassified by decision boundary function) since it is small and apart from big obstacle region, become meaningful obstacle region decided by decision boundary Eq. (5.2.10) as the value of σ decreases. In general small σ makes SVM overfitted to given data. This is not recommended for data classification because overfitted classifier does not work well to new data. In this problem, however, mapping from latent space to workspace is not considered as noise corrupted function. Thus, overfitting improves performance to capture the shape of obstacles in latent space. Red and blue points are misclassified samples. Reds are misclassified as obstacle and blues are the opposite cases. As σ decreases, misclassified region decreases towards decision boundary. Statistics in Table 5.2 also supports these result. (As σ decreases, $dist_{latent}$ decreases.) As value of σ decreases, not only classifier has small cross validation error e_{cv} with training data $\mathcal{D}_{train}$, but also error e_{test} of test data $\mathcal{D}_{test}$ is reduced.

Distance from decision boundary in data space, i.e. in latent space, cannot be obtained directly while distance in feature space is easy. Distance $dist_{latent}$ from decision boundary can be obtained by following optimization, for a given misclassified sample x'

$$L = \min_{x} \frac{1}{2}||x - x'||^2 \quad \text{s.t.} \quad B^2(x) = 0. \tag{5.2.12}$$

As shown in Table 5.2, if $\sigma = 0.1$, average distance $dist_{latent}$ between decision boundary and misclassified samples from $\mathcal{D}_{test}$ is less than average sampling distance $||\delta x||_1$ computed from $\mathcal{D}_{train}$. It means that most of the misclassified samples are within the average sampling interval. Thus, most of them are not sampled with average sampling distance (5.2.8). That is, if the value of σ is small enough, the

obstacle can be mapped to the latent space to a desired level with a small amount of data. The SVM learning time is also reduced by a small amount of data.

Table 5.2: SVM Statistics

	e_{learn} (%)	e_{cv} (%)	e_{test} (%)	$dist_{latent}$	$dist_{feature}$	$\|\|\delta x\|\|_1$
$\sigma = 1$	6.04 ± 2.76	6.30 ± 2.78	6.10 ± 2.80	0.21 ± 0.075	0.010 ± 0.0020	
$\sigma = 0.5$	1.74 ± 1.19	2.17 ± 1.20	1.85 ± 1.20	0.095 ± 0.096	0.0067 ± 0.0031	0.0066 ± 0.0023
$\sigma = 0.1$	0.038 ± 0.019	1.19 ± 0.27	0.63 ± 0.092	0.0053 ± 0.0013	0.0053 ± 0.0011	

5.2.4 Motion Planning in Latent Space

As mentioned earlier obstacles in the observation space are mapped into a highly complex shape in the latent space. We used VF-RRT (Vector Field Rapidly-Exploiting Random Tree) [51] [52] to plan a new path in this environment. Although generic RRT methods ensure that they generate motion paths that avoid obstacles in any complex obstacle environment, they are not suited for learning from demonstration that takes account of accumulated work knowledge because of the inherent randomness of the algorithm. However, if one generates an appropriate vector field that reflects the accumulated knowledge of the work, one can apply VF-RRT to learning from demonstration. This is because the VF-RRT has the property of extending the search tree in such a way as to generate random nodes along the direction of a given vector field.

5.2.4.1 Vector Field RRT

In previous literature [51, 52], the authors defined an upstream criterion to quantitatively measure the difference between a path moving along the direction of a vector field and a path moving against a given vector field. This quantitative measure is given as a functional form of path integrals and can be obtained as a direct consequence of the Cauchy-Schwarz inequality. For given unit-speed path $q : [0, L] \to \mathcal{Q}_{free}$ and piecewise continuous vector field $f : \mathcal{Q} \to \mathcal{TQ}$, upstream criterion $\mathcal{U}(q)$ is defined as follows:

$$\mathcal{U}(q) = \int_0^L \{||f(q(s))|| - \langle f(q(s)), q'(s) \rangle\} ds, \tag{5.2.13}$$

where $\mathcal{Q} \subseteq \mathbb{R}^n$ is $n-$dimensional configuration space manifold in $n-$dimensional Euclidean space, $\mathcal{Q}_{free}$ is a free space, $\mathcal{TQ}$ is a tangent space of $\mathcal{Q}$, and $|| \cdot ||$ is the norm induced from the inner product $< \cdot, \cdot >$. The greatest property of

the minimum upstream path is that it can be regarded as the minimum control effort path. The new node in the RRT tree can be determined by a combination of random sampled directions and vector field effects. A weighting factor of the vector field is sampled from a probability density function, which is closely related to the upstream criterion. In this probability density function, the lower the upstream cost, the higher the probability. For the conservative vector field, the following potential function $V(q)$ exists

$$f(q) = -\nabla V(q). \tag{5.2.14}$$

It is also applicable to VF-RRT when the potential function is continuous or the piecewise $\mathcal{C}^1$ function. For further properties and mathematical details, we refer to the reader [51], [52].

5.2.4.2 Potential Function for VF-RRT

We have tried to use the similarity measure in latent space between learned motion and newly generated motion as a potential function for VF-RRT. If a potential function is defined to have such characteristics, the VF-RRT can generate a motion similar to the learned motion. Because it is difficult to reflect physical properties in the GPDM latent space, a similarity measure is defined as follows using the kernel function of GPDM latent space.

$$V(x) = -\left(\mathbf{1}_{N-1}^T K_X^{-1} k_X(x)\right)^2, \tag{5.2.15}$$

where $\mathbf{1}_{N-1}$ is $N-1$ dimensional vector consisting of all elements of 1, $X = [x_1, \cdots, x_N]^T \in \mathbb{R}^{N\times d}$, $K_X(x) \in \mathbb{R}^{N-1\times N-1}$ is a kernel matrix whose elements $(K_X)_{ij}$ are $k_X(x_i, x_j)$, $i, j = 1, \cdots, N-1$, and $k_X(x) = [k_X(x_1, x), \cdots, k_X(x_{N-1}, x)]^T$.

The feature of this similarity measure is that the cost of the learned motion is the lowest as seen in the Figure 5.4a, and the cost around it increases monotonically as it moves away from the learned motion. The vector field in the GPDM latent space can be obtained by differentiating the above potential function Eq. (5.2.15):

$$\nabla V(x) = -2\left(\mathbf{1}_{N-1}^T K_X^{-1} k_X(x)\right)\left(\mathbf{1}_{N-1}^T K_X^{-1} \frac{\partial k_X(x)}{\partial x}\right). \tag{5.2.16}$$

The proposed potential map and its vector field are shown in Figure 5.4. The feature of our proposed vector field Eq. (5.2.16) is similar to the dynamics of GPDM latent space. The dynamics in the GPDM latent space can be expressed as the most probable next state Eq. (2.2.40) in the probability density function Eq. (2.2.39).

$$\mu_X(x) \quad = \quad X_{out}^T K_X^{-1} k_X(x)$$

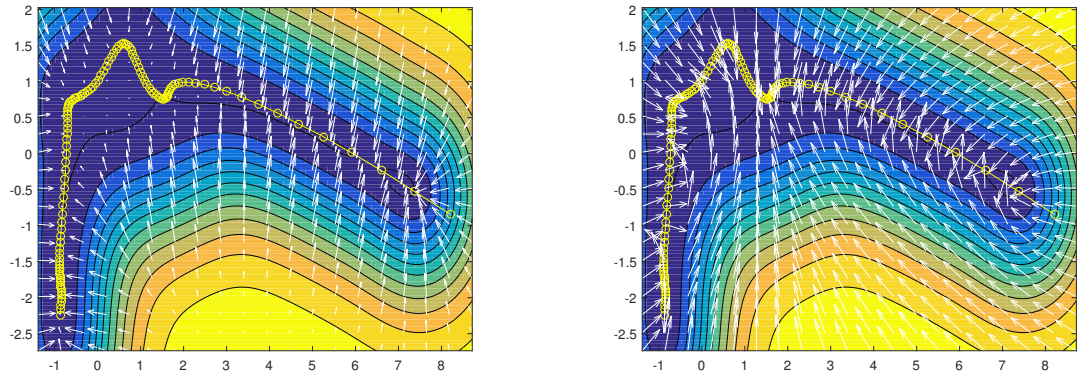
where $k_X(x) = [k_X(x_1, x), k_X(x_2, x), \cdots, k_X(x_{N-1}, x)]^T \in \mathbb{R}^{N-1}$, I_d is d-dimensional identity matrix and $X_{out} = x_{t2}^N$ for the first-order Markov process assumption. In the Figure 5.4b, each vector indicates the direction of the most probable next latent space coordinate from each starting point by the dynamics in the latent space by GPDM learning. The dynamics in the GPDM latent space converges to the learned motion direction. On the latent path, it has a direction from the starting point to the goal point.

If the learned motion data has multiple paths, the potential function can be defined as continuous and piecewise $\mathcal{C}^1$ function:

$$V_i(x) \quad = \quad -\left(\mathbf{1}_{N_i-1}^T K_{X_i}^{-1} k_{X_i}(x)\right)^2 \tag{5.2.17}$$

$$V(x) \quad = \quad \min V_i(x), \tag{5.2.18}$$

where $X = [X_1^T, \cdots, X_m^T]^T$, m is number of learned motions, $X_i \in \mathbb{R}^{N_i \times d}$, $K_{X_i} =$



(a) (b)

Figure 5.4: Potential map and vector field for a learned motion in 2-dimensional latent space. Yellow trajectory is a learned motion. (a) Proposed potential map and vector field. (b) Dynamics in latent space.

$K_X(X^i_{in}, X^i_{in})$, $k_{X_i}(x) = k_X(X^i_{in}, x)$, $i = 1, \cdots, m$, and $X^i_{in} = [x^i_1, \cdots, x^i_{N_i-1}]^T \in \mathbb{R}^{(N_i-1)\times d}$.

Figure 5.5 is a potential map of the four learned motions. It can be seen that the cost of the learned motions, as in Figure 5.5, has the smallest value and increases monotonically as it moves away from these motions. One of the characteristics of VF-RRT is that it generates a new path in the potential field with a small potential value. Since we have designed the learned motion to have the smallest potential value in the latent space, the VF-RRT generates a new path similar to the learned motion. Therefore, VF-RRT generates a path that is much more natural than other RRT-based algorithms. By comparing the cost of the path generated by the basic RRT and VF-RRT, we can see which algorithm is similar to

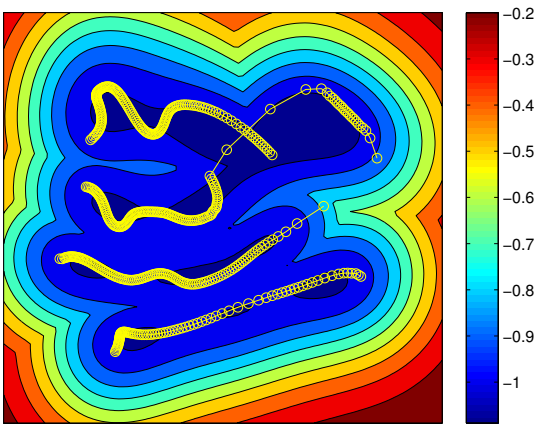
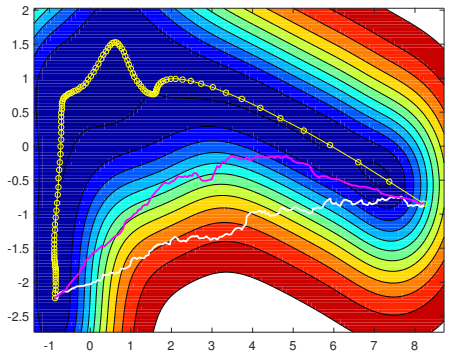
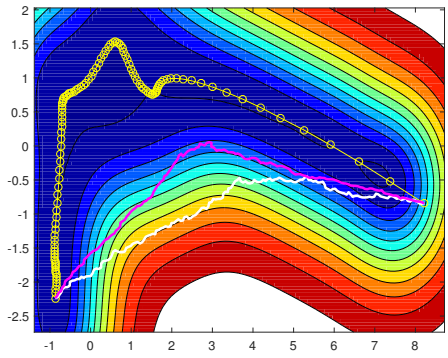


Figure 5.5: Potential map and four learned motions in latent space.

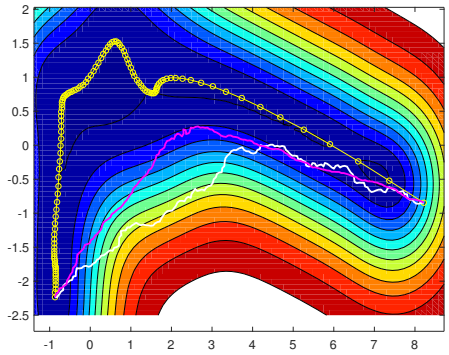
the learned motion. The VF-RRT also has a user-defined parameter called reference exploration efficiency factor, $E_s =\in (0, 1)$. The closer this value is to one, the greater the tendency to generate a path to a lower cost. The comparison of experimental results of RRT and VF-RRT for fixed stepsize is summarized in Table 5.3 and Figure 5.6. As shown in Figure 5.6, the existing RRT shows a straight line connecting the starting point and the goal point in an obstacle-free environment. On the other hand, in the case of VF-RRT, it can be seen that as E_s increases, it generates a path that is more similar to the learned path. This can also be seen by comparing the cost values of Table 5.3. Furthermore, it can be seen that the length of the generated path is shorter and the number of iterations is smaller even though the path of the VF-RRT is not a straight path. On the other hand, the efficiency of the VF-RRT is low in terms of calculation time.



(a) $E_s = 0.15$



(b) $E_s = 0.25$



(c) $E_s = 0.45$

Figure 5.6: Effects of E_s. Red line is the resulting path of VF-RRT and white line is the resulting path of basic RRT. As E_s increases, generated path is more close to the learned path.

Table 5.3: Comparing VF-RRT with RRT

	Base	RRT	VF-RRT		
			$E_s = 0.15$	$E_s = 0.25$	$E_s = 0.45$
Cost	-0.9983	-0.7855	-0.8518	-0.8714	-0.8823
Std. of Cost	0.0177	0.1470	0.0964	0.0845	0.0784
No. Iteration	-	152.0	132.1	134.9	143
Path Length	-	182.7	175.1	176.8	178
Time (sec)	-	0.0701	0.3951	0.4152	0.4176

5.3 Experiments

5.3.1 Grasping Experiments

In this experiment, we performed motion capture of human demonstration motion through Kinect and CyberGlove II and then learned the robot. For a human demonstration motion Y and the corresponding robot motion Z, which is obtained by solving inverse kinematics of Y, we learned GPDM so that Y and Z have the same latent space. In this section, we have applied the learning by demonstration framework introduced in the previous sections to learn GPDM latent space and then applied to the grasping motion of an arm of a humanoid.

To teach robots from human demonstrations, motion was captured using Kinect and CyberGlove II. The dimensions of the data obtained using these instruments were 26, 7 for the arm, and 19 for the hand. Using Kinect's human motion tracking function, five joint angles can be obtained from the shoulder joints, one joint from

Figure 5.7: The MAHRU model.

elbow's flexion and extension, and one joint from wrist's adduction and abduction. The wrist joint values (pronation/supination) for twisting the forearm were not measurable by Kinect, so data were collected while the palm was fixed to the side view. In other words, the data was collected by constraining the normal vector direction of the palm to be included in the transverse plane of the anatomical plane of the human body. The remaining 19 joint data were collected from CyberGlove II and consisted of fingers and wrist motion except for pronation/supination.

The robot platform used in this experiment is a humanoid robot Mahru that was developed by KIST (Korea Institute of Science and Technology). A manipulator of the Mahru consists of a 7-DOF arm and a 3-DOF ROBOTIS PHN-33B hand with three fingers.

Now user demonstrates the action of grasping a cylindrical object on a table. Then the corresponding robot motion can be obtained by applying constraints to grasp the same object placed at the same position and solving inverse kinematics. For this experiment, we demonstrated the motion of grasping the same object in four different places. Then we solved the inverse kinematics and got the motions

Table 5.4: The specification of the MAHRU

Dimension	600×1500 mm
Weight	67 kg
DOF of whole body	35
DOF of manipulator	10(arm $7 +$ hand 3)

of the robot. The final postures of the learned movements are shown in Figure 5.8.

We have learned GPDM so that Y and Z share the same latent space for human demonstration motion Y and its corresponding robot motion Z. Details on learning GPDM when given multiple motions are given in [2]. Back-constraints [53] are given between robot motion and latent space. We trained GPDM that the inverse mapping from the robot joint space to the latent space is smooth, so that the final posture of the robot, which can grasp the same object located at slightly different places, can be well mapped to the latent space.

The experimental environment is shown in Figure 5.9. There is a table (big red box) and a book is placed on it. The demonstration environment of the learned movements assumes that the edges of the table and the book are aligned so that the arm of the robot does not collide with the edge of the table or the book. In this experiment, the book protrudes $60mm$ from the edge of the table and is regarded as an obstacle. The target cylindrical object ($\phi 50 \times 165mm$) is on the book.

The new final posture of the robot to grasp the same object in a position slightly different from the final position of the learned motion was found by solving the inverse kinematics. To make easier the problem, we assumed that the gripping posture and gripping position are the same as one of the postures of the learned

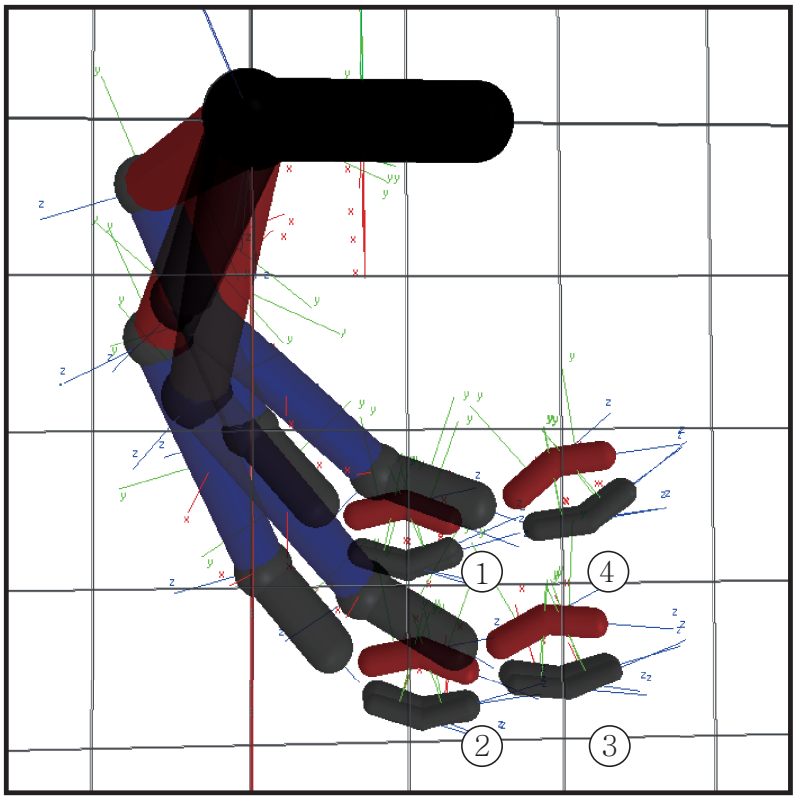


Figure 5.8: Four final poses of learned sequences.

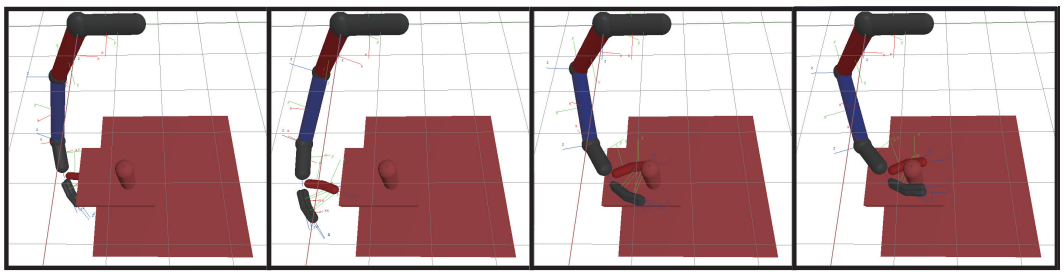


Figure 5.9: Simulation results of grasp motion.

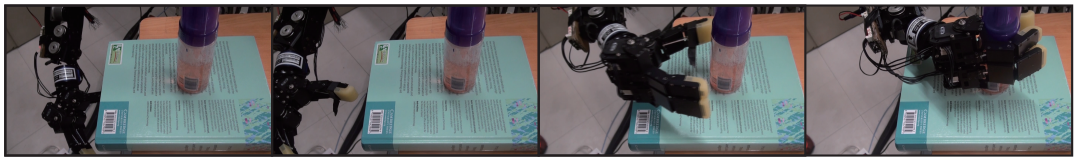


Figure 5.10: Experiment results of grasp motion.

Table 5.5: Sampling Distances and Numbers of Sampling

# of data	114 (1 motion)	473 (4 motions)
c mean ± std.	5.82 ± 2.08	28.61± 4.28
sampling distance $\frac{1}{c} \times 0.025$	4.3×10^{-3}	8.74×10^{-4}
# of sampling	2.02×10^{6}	4.88×10^{7}

motion. Using grasp planner such as GraspIt!, one can find other grasping positions and grasping postures. After solving the inverse kinematics and finding the new final position of the robot in joint space, GPR was used to find the coordinates in the latent space. Also, table and object are mapped to latent space using general SVM. The sampling distance to avoid an obstacle of about 1 inch was calculated through the formula Eq. (5.2.8). The number of samples needed for the latent space area ($[-2.4, 4.3] \times [-2.9, 2.6]$ latent space) of Figure 5.11 is summarized in Table 5.5. The arm length of the robot Mahru was assumed to be $1m$. As mentioned above, the value of c increases as the number of data increases.

Finally, VF-RRT was used to generate a grasping motion avoiding collision with obstacles and target objects. The path planning results of VF-RRT are shown in Figure 5.11. The generated path was manually smoothed using B-spline in order

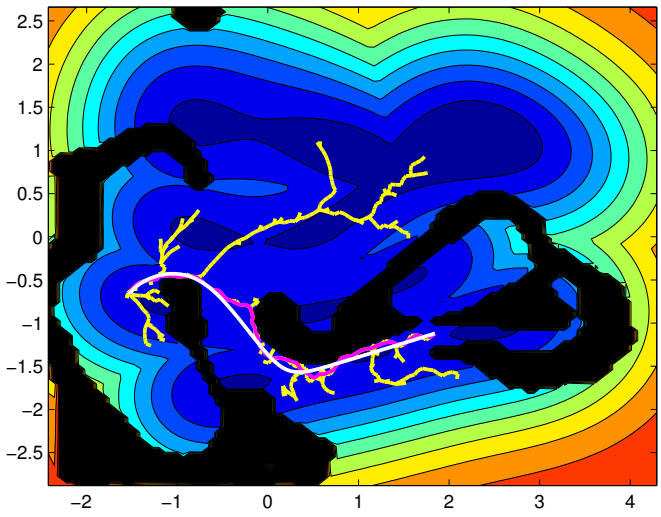


Figure 5.11: Planned path using VF-RRT in latent space. Black regions are obstacles and constraints. Yellow tree is the result of VF-RRT and magenta path is the shortcut path. White path is the manually smoothed path with B-splines.

to apply it to the actual robot, and it was not optimized for the kinematics and dynamics of the robot. The reason for this is to show the efficiency of VF-RRT in latent space using the potential field formula Eq. (5.2.15) defined using the GPDM kernel function. The robot dynamics were considered to some extent without optimization. Experimental results using the simulator and the robot platform are shown in Figure 5.9 and Figure 5.10, respectively.

5.4 Conclusion

In this chapter, we have examined whether the Gaussian process dynamical model can be extended to one of the dimensional reduction methods of the learning from demonstration. In the large frame, we proposed a learning from demonstration framework using GPDM. A method of mapping the obstacles in the workspace to the latent space using SVM is proposed. In addition, a criterion is provided for the sampling distance to avoid missing objects of the specified size. The data for SVM learning was obtained through uniform sampling in the latent space. The corresponding label of the data was given after checking whether it collided with an obstacle in the workspace. The uniform sampling distance was determined by using the proposition derived from the forward kinematics of open chain robot. In order to test the performance of our framework, we considered the manipulator to perform grasping while avoiding obstacles. In order to plan the obstacle avoidance path, a potential function reflecting knowledge of the learned motions in the latent space was defined and a new collision-free path was generated by using it in VF-RRT. In this study, we focused on the method of bringing the obstacles into the latent space when there are obstacles in the workspace, and the motion planning method reflecting the learned knowledge in the latent space through GPDM

learning.

6

Conclusion

This thesis has addressed the problem of planning and recognizing motion intention for high-dimensional robot systems operating in various environments which are obstacles that should be avoided or terrains that should be identified. In order to address those problems, we exploit the forward kinematics expressed as the product of exponentials and the Gaussian process dynamical models (GPDM) . We conclude this thesis with the following summary.

- **GPDM-Based Human Motion Intention Recognition for Lower-Limb Exoskeleton**

  We have proposed a real-time method of recognizing lower-limb exoskeleton robot wearer's motion intention depending on the terrain on which the wearer is moving based on GPDM. The method is to compare logarithms of conditional probability distribution of new observation and the corresponding GPDM latent variables given training data of a certain motion type. Although there are many kinds of sensors and their signals are corrupted by

different levels of noise, GPDM captures dynamic characteristics of a motion well. So our method recognizes a wearer's motion intention well than other machine learning-based classifiers such as LDA, SVM, and an HMM-based algorithm. One of the advantages of our algorithm is that it produces pretty good results without the EMG or EEG signals that are used in most previous studies. Proposed method is validated through two kinds of data sets: joint angle paths which are calculated from 3D motion tracking system attached to human body parts and sensor signals including joint angles, AHRS, and IMUs installed in a lower-limb exoskeleton robot.

- **An Adaptive Stepsize RRT Planning Algorithm for Open Chain Robots**:

  We have proposed an adaptive stepsize RRT path planning algorithm for open chain robots. Our algorithm guarantees that the maximal deviations of the points on each of the robot's links are bounded by a minimum obstacle size which is the only input parameter. Using a standard operator norm inequality as well as the structure of an open chain's forward kinematics equations as captured by the product of exponentials formula, we derive an approximate bound on the Cartesian displacement of the open chain tip for a given joint space displacement. We extend this bound to take into account the points of each link which is modeled as a convex polytope. The extension of the bound determines an allowable stepsize at each iteration. Not having to manually set the stepsize parameter through a time-consuming trial-and-error process is an important feature of our algorithm.

- **A Gaussian Process Dynamical Model-Based Planning Method**

  We have proposed a motion planning method that generates a collision-free path and a motion similar to the learned motion in GPDM latent space.

Dimensionality of a demonstrated motion is reduced by GPDM. Using the bound on the Cartesian displacement proposed in our previous research, an appropriate sampling resolution for mapping obstacles into the latent space by SVM is determined. We show that an appropriate RBF kernel parameter should be determined properly in order to map out well regions corresponding to collisions. Then the appropriate RBF kernel parameter can reduce the required number of samples and thus allow the SVM to be trained more efficiently. To measure the similarity between the learned motion and a newly generated motion we define a potential function using GPDM kernel function. Using this potential function, VF-RRT generates a path that avoids obstacles and is similar to the learned motion.

There are several ways in which the above studies can be developed more meaningfully. First, if a study of the motion intention estimation of lower-limb exoskeleton wearer can predict not only the current motion intention but also how the wearer will move in the near future, the exoskeleton robot can more effectively assist the wearer's motion. To estimate multiple steps ahead, a higher order Markov model may be needed that uses more observation data than the current second order Markov model. In this case, however, the complexity of the model will make it difficult to learn GPDM. Additionally, if we can exploit the zero velocity update (ZUPT), which is mainly used for indoor navigation [54, 55], together with the kinematic information of the lower-limb exoskeleton robot, we can predict wearer's next movements more accurately.

Second is to extend the bound on the Cartesian displacement of the open chain to closed loops. Compared to serial manipulators, parallel manipulators are widely used because of their high rigidity, high precision, and high speed. Also, when

human and robot collaborate, there is a high possibility that closed loop is formed between human and robot. In this case, it would be a meaningful study to extend the above bound to a closed loop for avoiding obstacles as well as safety reasons. This requires consideration of passive joints, redundancy, and constraints caused by loop-closures.

Bibliography

[1] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models. In *In NIPS*, pages 1441–1448. MIT Press, 2006.

[2] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):283–298, 2008.

[3] Christian Fleischer, Christian Reinicke, and Günter Hommel. Predicting the intended motion with emg signals for an exoskeleton orthosis controller. In *Intelligent robots and systems, 2005.(IROS 2005). 2005 IEEE/RSJ international conference on*, pages 2029–2034. IEEE, 2005.

[4] Ching-An Cheng, Tzu-Hao Huang, and Han-Pang Huang. Bayesian human intention estimator for exoskeleton system. In *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*, pages 465–470. IEEE, 2013.

[5] Mervin Chandrapal, XiaoQi Chen, Wenhui Wang, Benjamin Stanke, and Nicolas Le Pape. Preliminary evaluation of intelligent intention estimation algorithms for an actuated lower-limb exoskeleton. *International Journal of Advanced Robotic Systems*, 10, 2013.

[6] Huseyin Atakan Varol, Frank Sup, and Michael Goldfarb. Multiclass real-time intent recognition of a powered lower limb prosthesis. *Biomedical Engineering, IEEE Transactions on*, 57(3):542–551, 2010.

[7] He Huang, Fan Zhang, Levi J Hargrove, Zhi Dou, Daniel R Rogers, and Kevin B Englehart. Continuous locomotion-mode identification for prosthetic legs based on neuromuscular–mechanical fusion. *Biomedical Engineering, IEEE Transactions on*, 58(10):2867–2875, 2011.

[8] Hui He, Kazuo Kiguchi, and Etsuo Horikawa. A study on lower-limb muscle activities during daily lower-limb motions. *International Journal of Bioelectromagnetism*, 9(2):79–84, 2007.

[9] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Comput. Sci. Dept., Iowa State Univ., 1998.

[10] S. M. LaValle. Motion planning. *IEEE Robot. Automat. Mag.*, 18(1):79–89, March 2011.

[11] S. M. LaValle. Motion planning. *IEEE Robot. Automat. Mag.*, 18(2):108–118, 2011.

[12] M. Elbanhawi and M. Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014.

[13] J. J. Kuffner and S. M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.

[14] Ioan A Şucan and Lydia E Kavraki. On the implementation of single-query sampling-based motion planners. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2005–2011. IEEE, 2010.

[15] Chaoqun Wang and Max Q-H Meng. Variant step size RRT: An efficient path planner for uav in complex environments. In *Real-time Computing and Robotics (RCAR), IEEE International Conference on*, pages 555–560. IEEE, 2016.

[16] Michael J McCourt, Chau T Ton, Siddhartha S Mehta, and J Willard Curtis. Adaptive step-length RRT algorithm for improved coverage. In *AIAA Guidance, Navigation, and Control Conference*, page 0638, 2016.

[17] Roger W. Brockett. Robotic manipulators and the product of exponentials formula. In *Proc. Int. Symp. Math. Theory of Networks and Syst.*, pages 120–129. Springer, 1984.

[18] Tetsunari Inamura, Iwaki Toshima, and Yoshihiko Nakamura. Acquiring motion elements for bidirectional computation of motion recognition and generation. *Experimental robotics viii*, pages 372–381, 2003.

[19] Tetsunari Inamura, Naoki Kojo, Tomoyuki Sonoda, Kazuyuki Sakamoto, Kei Okada, and Masayuki Inaba. Intent imitation using wearable motion capturing system with on-line teaching of task attention. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 469–474. IEEE, 2005.

[20] Sylvain Calinon and Aude Billard. Stochastic gesture production and recognition model for a humanoid robot. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2769–2774. IEEE, 2004.

[21] Wataru Takano, Katsu Yamane, Tomomichi Sugihara, Kou Yamamoto, and Yoshihiko Nakamura. Primitive communication based on motion recognition

and generation with hierarchical mimesis model. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3602–3609. IEEE, 2006.

[22] Aaron P Shon, Joshua J Storz, and Rajesh PN Rao. Towards a real-time bayesian imitation system for a humanoid robot. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2847–2852. IEEE, 2007.

[23] Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[24] N.D. Lawrence. Gaussian process latent variable models for visualization of high dimensional data. *Advances in Neural Information Processing Systems*, 16:329–336, 2004.

[25] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[26] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[27] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[28] K. Grochow, S.L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 522–531. ACM, 2004.

[29] K. Yamane, Y. Ariki, and J. Hodgins. Animating non-humanoid characters with human motion data. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 169–178. ACM, 2010.

[30] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[31] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[32] Aaron M Dollar and Hugh Herr. Lower extremity exoskeletons and active orthoses: challenges and state-of-the-art. *IEEE Transactions on robotics*, 24(1):144–158, 2008.

[33] Adam B Zoss, Hami Kazerooni, and Andrew Chu. Biomechanical design of the berkeley lower extremity exoskeleton (bleex). *IEEE/ASME Transactions On Mechatronics*, 11(2):128–138, 2006.

[34] Kazuo Kiguchi, Takakazu Tanaka, and Toshio Fukuda. Neuro-fuzzy control of a robotic exoskeleton with emg signals. *IEEE Transactions on fuzzy systems*, 12(4):481–490, 2004.

[35] Tommaso Lenzi, Stefano Marco Maria De Rossi, Nicola Vitiello, and Maria Chiara Carrozza. Intention-based emg control for powered exoskeletons. *IEEE transactions on biomedical engineering*, 59(8):2180–2190, 2012.

[36] Raquel Urtasun, David J. Fleet, and Pascal Fua. 3d people tracking with gaussian process dynamical models. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 1:238–245, 2006.

[37] K. Yamane and Y. Nakamura. National motion animation through constraining and deconstraing at will. *Transactions on Visualization and Computer Graphics*, 9(3):353–360, 2003.

[38] Christopher J Leggetter and Philip C Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech & Language*, 9(2):171–185, 1995.

[39] John B Conway. *A course in functional analysis*, volume 96. Springer Science & Business Media, 1990.

[40] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. 2000.

[41] Jiří Rohn. Computing the norm $\|A\|_{\infty,1}$ is NP-hard. *Linear and Multilinear Algebra*, 47(3):195–204, 2000.

[42] Joel Aaron Tropp. *Topics in sparse approximation*. PhD thesis, Univ. Texas, Austin, 2004.

[43] Sylvain Calinon and Aude Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262. ACM, 2007.

[44] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 233. ACM, 2007.

[45] Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Nips*, volume 2, page 5, 2003.

[46] Aaron Shon, Keith Grochow, Aaron Hertzmann, and Rajesh P Rao. Learning shared latent structure for image synthesis and robotic imitation. In *Proc. NIPS*, pages 1233–1240, 2005.

[47] Raquel Urtasun, David J Fleet, Andreas Geiger, Jovan Popović, Trevor J Darrell, and Neil D Lawrence. Topologically-constrained latent variable models. In *Proceedings of the 25th international conference on Machine learning*, pages 1080–1087. ACM, 2008.

[48] Hyuk Kang and FC Park. Motion optimization using gaussian process dynamical models. *Multibody System Dynamics*, pages 1–19, 2014.

[49] H. Kang and F. C. Park. Humanoid motion optimization via nonlinear dimension reduction. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1444–1449, 2012.

[50] Alex J Smola and Bernhard Schölkopf. *Learning with kernels*. Citeseer, 1998.

[51] Inyoung Ko, Beobkyoon Kim, and Frank Chongwoo Park. VF-RRT: Introducing optimization into randomized motion planning. In *Proc. Asian Control Conference*, pages 1–5, 2013.

[52] Inyoung Ko, Beobkyoon Kim, and Frank Chongwoo Park. Randomized path planning on vector fields. *Int. J. Robotics Research*, 2014.

[53] Neil D. Lawrence. Local distance preservation in the gp-lvm through back constraints. In *In ICML*, pages 513–520. ACM Press, 2006.

[54] Sang Kyeong Park and Young Soo Suh. A zero velocity detection algorithm using inertial sensors for pedestrian navigation systems. *Sensors*, 10(10):9163–9178, 2010.

[55] Min Su Lee, Hojin Ju, Jin Woo Song, and Chan Gook Park. Kinematic model-based pedestrian dead reckoning for heading correction and lower body motion tracking. *Sensors*, 15(11):28129–28153, 2015.

국문초록

본 논문에서는 로봇이 해석적으로 정의되지 않은 환경에 대응하는 문제에 관해 다룬다. 이 환경에는 로봇이 피해야 하는 장애물과 하지 외골격 로봇 착용자의 동작 의도와 밀접하게 연관된 지형지물이 있다. 관절 공간과 그 저차원 공간에서의 경로 계획법을 통해 장애물을 회피 하였고 기계학습 기법을 통해 지형지물에 기인한 사람의 동작 의도를 추정하였다.

먼저 Gaussian process dynamical models (GPDM) 기반으로 하지 외골격 로봇 착용자의 운동 의도를 추정하는 알고리즘을 제안하였다. 관측한 짧은 시계열 입력 값에 대하여 이에 상응하는 저차원 공간 좌표를 Gaussian process regression 을 통해 얻는다. 각 모델에 대한 유사도는 학습 데이터에 대한 관측 값과 그 저차원 공간 좌표의 로그 조건부 확률분포 형태로 표현된다. 이 유사도를 비교하여 가장 가능성 있는 동작을 추정한다. 하지 외골격 로봇 프로토타입 및 동작 추적 시스템을 이용한 물리적 실험을 통해 우리의 알고리즘을 검증하였다.

다음으로는 적응적으로 스텝사이즈를 결정하는 RRT 알고리즘을 제안하였다. 지수 곱(Product of Exponentials, PoE) 형태로 표현된 로봇의 정기구학과 표준 작용소 노름 부등식으로부터 직렬 개 연쇄 로봇의 엔드이펙터의 작업공간에서의 최대 변위와 관절 공간에서의 변위에 대한 부등식으로 유도하였다. 이 부등식을 이용하여 주어진 장애물의 최소 크기에 대하여 적응적으로 스텝사이즈를 결정하였다. 10 자유도 평면 개 연쇄 로봇과 7축 산업용 매니퓰레이터를 이용하여 우리의 알고리즘을 검증하였다.

마지막으로 사람의 시연 동작을 GPDM을 이용해 저차원 공간으로 학습하여, 사람과 유사한 동작을 생성하는 저차원 공간에서의 경로 계획 방법을 제안하였다. 앞서 유도한 부등식을 저차원 공간에서의 변위와 작업공간에서의 각 링크의 변위에 대한 부등식으로 확장하였다. 이를 이용하여 작업공간에서 정의된 장애물을 샘플링 기반으로 저차원 공간으로 매핑하였다. 그리고 학습한 동작과 새롭게 생성한 동작 사이의 유사성을 측정하는 측도를 GPDM 커널함수를 기반으로 정의하였다. 시뮬레이터와

실제 로봇에 적용해 봄으로써 제안한 방법의 유효성을 검증하였다.

주요어: 가우시안 프로세스 동적 모델, 경로 계획, 잠재 공간에서의 경로 계획, 동작 의도 인식, 기계학습, rapidly-exploring random tree, 적응적 스텝사이즈, 작용소 노름.

학번: 2009-20694