A Programming by Demonstration with Least Square Support Vector Machine for Manipulators

Jingdong Zhao, Chongyang Li, Zainan Jiang(✉), and Hong Liu

State Key Laboratory of Robotics and System,
Harbin Institute of Technology, Harbin 150001, China
jiangzainan@hit.edu.cn

Abstract. This paper presents a method of programming by demonstration, aiming at instructing the manipulator to accomplish tasks with obstacles in the way or with strict motion paths. Least square support vector machine (LS-SVM), based on the principle of structure risk minimization, is employed to achieve better generalization and reproduced trajectories with higher accuracy. Furthermore, the velocity field method is applied to maintain the convergence of reproduced trajectories and smooth the motion. Finally, a series of obstacle avoidance experiments with a 7-DOF manipulator are conducted to verify the feasibility of the proposed method.

Keywords: Programming by demonstration · LS-SVM · Velocity field method · Obstacle avoidance

1 Introduction

In recent years, with the advance of robotics and their widespread applications, manipulating tasks have become more and more complex. Manual programming, which demands the expertise of the programmer and could hardly adjust to the environment, is tedious, and more and more difficult to meet the requirements from changeable occasions. Thus, there arises the demand of the manipulator trajectory planning method, which is able to operate conveniently and adapt to the environment.

Programming by demonstration (PbD) is an effective solution for the last issue [1-3]. This method endows the manipulator with the ability to learn demonstrations from human operators. In PbD, some most commonly used methods for trajectory planning are as follows: methods based on statistics [4-6], methods based on dynamical system (DS)[7-8], methods based on spline[9].

Many researchers modeled and reproduced trajectories employing the statistics-based methods. For example, Tso [4] modeled and reproduced the motion of the human by Hidden Markov Models (HMM); Calinon [5-6] encoded a series of trajectories by Gaussian Mixture Models (GMM), and reproduced the trajectories in different environments using Gaussian Mixture Regression (GMR). Ijspeert [7] firstly proposed encoding the motion by DS in the form of nonlinear differential equations. Learning algorithms can be integrated into the framework of DS, such as GMM [10] and locally weight regression (LWR) [11].

H. Liu et al. (Eds.): ICIRA 2015, Part III, LNAI 9246, pp. 126–137, 2015.
DOI: 10.1007/978-3-319-22873-0_12

However, the above methods mainly focused on point-to-point motions, rather than the obstacle avoidance during tasks. In this paper, we present a new method of PbD. With the DS as its framework, this method uses least square support vector machine (LS-SVM) to encode the motion and adds the velocity field method nearby the target position. This paper completes the theoretical derivation and analysis of this method, and validates the feasibility by experiments.

2 Overview

Fig. 1 shows the architecture of the manipulator system, using PbD as its foundation.

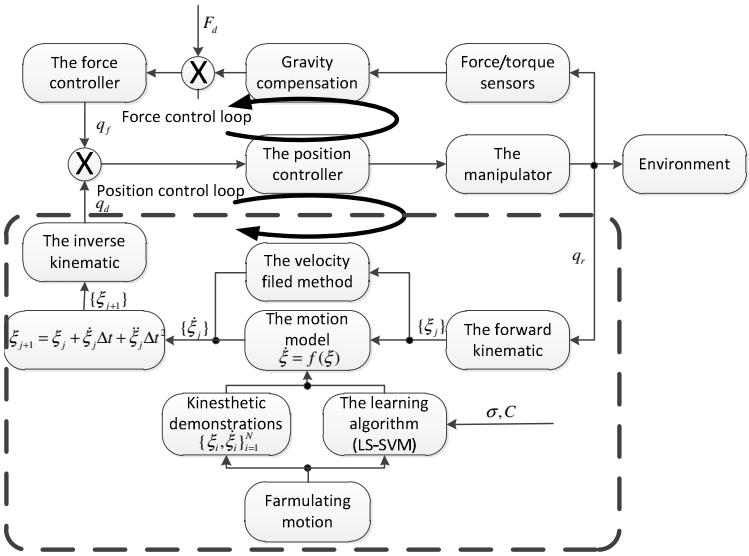

Fig. 1. The manipulator system

The manipulator system was composed of two closed loop: the position control loop and the force control loop. For protecting the manipulator, the force control loop corrected the desired joint position of the manipulator when it received the supernumerary force from the environment. The LS-SVM-based PbD was applied in the position control loop, which reproduced the trajectory of the task by regression. The whole work was done in two phases.

Phase 1: a global motion model, formulated by recursive positions or DS principle respectively, with more details illustrated in Section 2.1, was established based on the principle of LS-SVM with the information from kinesthetic demonstrations.

Phase 2: the trajectory was reproduced from its initial. By recalling the global motion model iteratively, the trajectory would converge to the end position of the task.

2.1 Formulism

Formulism Based on Position

This formulism for encoding the motion could be summarized as follows: the position of the next cycle was obtained in accordance with the current position by a certain conversion. In other words, it encoded the motion with the position-based recursive expression:

$$\xi_{i+1} = f(\xi_i) \tag{1}$$

$\xi_i \in R^n$ indicates the current position, while $\xi_{i+1} \in R^n$ indicates the next position of the trajectory. Each entry of the vector ξ_i represents the coordinate in the joint space or the Cartesian space.

Formalism Based on DS

Khansari-Zadeh [12] pointed out that the motion could be governed by an autonomous ordinary differential equation. By ignoring inaccuracies in sensor measurements and errors resulting from imperfect demonstrations, the equation would be:

$$\dot{\xi} = f(\xi) \tag{2}$$

Where $f : R^n \to R^n$ is a nonlinear continuous and continuously differentiable function with a single equilibrium point $\dot{\xi}^* = f(\xi^*) = 0$ .Different input state variables ξ correspond to different order of the equations (e.g., ξ could be the joint angles, the position of the end-effector, the position and velocity of the end-effector, etc.). According to the request for the precision of the position and other requests of the manipulator system, ξ expresses the position and velocity of the manipulator's end-effector in this paper.

The impact of these two ways to reproduce trajectories would be discussed through experiments in a later section.

2.2 Kinesthetic Demonstrations and the Learning Algorithm

In this paper, the kinesthetic demonstrations were the position and velocity of N given points on L trajectories. There were many methods to acquire the kinesthetic demonstrations, such as visual detecting and location tracking. In our research, human operator demonstrated the skill to the manipulator by dragging it on the model of the zero force control.

To acquire the formulated relationship by kinesthetic demonstrations, LS-SVM was introduced in this paper. This algorithm is based on the principle of structure risk minimization [13]. Compared with previous algorithms used in PbD (GMM, HMM and so on), LS-SVM has better generalization. And LS-SVM could be trained faster than the traditional SVM [14].

For the set of N given demonstrations on L trajectories $\{\xi_i, \dot{\xi}_i\}_{i=1}^N$, $\xi_i \in R^n$ represents the input variable, and $\dot{\xi}_i \in R^n$ indicates the desired output of ξ_i. The regression function of ξ is $f(\xi) = \mathbf{w}^T \varphi(\xi) + b$, and its LS-SVM-based regression problem can be transformed into an optimization problem with constraints:

$$\begin{cases} \min\limits_{\mathbf{w},b,\mathbf{e}} J_{LS} = \min\limits_{\mathbf{w},b,\mathbf{e}} (\frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum\limits_{i=1}^{N} e_i^2) \\ s.t. \quad y_i - [\mathbf{w}^T \varphi(\xi_i) + b] = e_i, i = 1, 2, ..., N \end{cases} \tag{3}$$

Where b defines the bias of the regression function, e_i is the training error of demonstration $\{\xi_i, \dot{\xi}_i\}$, and $\varphi(\bullet)$ indicates the mapping from demonstrations to kernel feature space. $C > 0$ is the penalty function, which plays an important role on balancing the generalization of algorithm and the tolerance error. The influence of C to the reproduced trajectories would be discussed in a later section. The output equation of LS-SVM is:

$$f(\xi) = \sum_{i=1}^{N} \alpha_i k(\xi_i, \xi) + b \tag{4}$$

$$\begin{cases} b = \dfrac{\mathbf{E}_V{}^T (\mathbf{K} + C^{-1}\mathbf{I})^{-1} \mathbf{Y}}{\mathbf{E}_V{}^T (\mathbf{K} + C^{-1}\mathbf{I})^{-1} \mathbf{E}_V} \\ \boldsymbol{\alpha} = (\mathbf{K} + C^{-1}\mathbf{I})^{-1} (\mathbf{Y} - b\mathbf{E}_V) \end{cases} \tag{5}$$

$\mathbf{E}_V = [1, 1, ..., 1]^T$, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_N]^T$, $\mathbf{Y} = [y_1, y_2, ..., y_N] \in R^{N \times 1}$, $\mathbf{K}, \mathbf{I} \in R^{N \times N}$, $\mathbf{I}$ is the unit matrix, and $\mathbf{K}$ is the kernel matrix. Our method used Gaussian kernel (6) to map the feature space to the infinite dimension.

$$\mathrm{K}_{i,j} = k(\xi_i, \xi_j) = \exp(-\frac{\left\| \xi_i - \xi_j \right\|^2}{\sigma^2}) \tag{6}$$

σ is the coverage of Gaussian kernel. And the effect of σ to the reproduced trajectories would be discussed in a later section.

Without the loss of generality, we take the constructing of DS model as an example to demonstrate the establishment of global motion models. Within the DS model, the kinesthetic demonstrations are $\{\xi_i, \dot{\xi}_i\}_{i=1}^N$. Wherein, the input variable $\xi_i = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$ is detected by the sensors of the manipulator, and $x \in R, y \in R, z \in R$ are the coordinates of the manipulator's end-effector in the Cartesian space, and $\dot{x} \in R, \dot{y} \in R, \dot{z} \in R$ are the velocity components. The output variable $\dot{\xi}_i = [\dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}]$ is calculated by $\dot{\xi}_i = (\xi_i - \xi_{i-1}) f_s$, and f_s indicates the sampling frequency of the sensors. In order to obtain the mapping from ξ to $\dot{\xi}$, kinesthetic demonstrations were used for training

the DS using LS-SVM. As the system had multiple-input and multiple-output (MIMO), the six output variables needed to be trained respectively. Then, the motion was represented as six output equations, such as:

$$\dot{x} = f_{\dot{x}}(\xi) = \sum_{i=1}^{N} \alpha_{\dot{x},i} k(\xi_i, \xi) + b_{\dot{x}} \tag{7}$$

In the light of the output equations, the position of the next time would be calculated using the current position and velocity of the manipulator, through the equation of motion along three directions, such as:

$$x_{i+1} = x_i + \dot{x}_i \Delta t + \frac{1}{2} \ddot{x}_i \Delta t^2 \tag{8}$$

Where Δt indicates the step time, which is the control period of the manipulator. However, if the running time of the manipulator bot needed to be changed, Δt could be increase or decrease appropriately. Each position on this trajectory was calculated according to the last position by this method.

2.3 Velocity Field Method

When demonstrating the manipulator to approach the target position precisely, the human operator needed to drag the manipulator repetitively to aim at the target point, which would result in the tremble of the reproduced trajectory at the target point.

To avoid the tremble, this paper presents a method based on the velocity field. The method was applied nearby the target position, to ensure the reproduced trajectory could converge smoothly and quickly. Fig. 2 shows the velocity field:

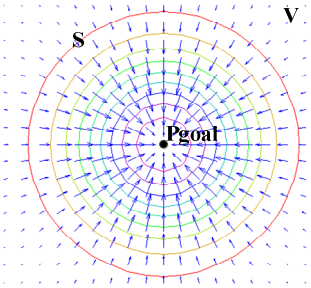

Fig. 2. The velocity field

Pgoal represents the target position, and S is a circle with the center Pgoal. The direction of the arrow indicates the normal direction of circle S, which represents the velocity direction of the current position. Arrow density is in inverse proportion to the velocity. Thus, the formula about the velocity in the velocity field could be obtained:

$$\mathbf{V} = \mathbf{k}(\mathbf{P} - \mathbf{P}_{goal}) \tag{9}$$

$\mathbf{V} \in R^3$ denotes the velocity on the position $\mathbf{P} \in R^3$. $\mathbf{k}$ is used to ensure the manipulator's velocity continuous when it enters the velocity field, which was obtained as follows:

$$\mathbf{k} = \frac{\mathbf{V}_{in}}{\mathbf{p}_{in} - \mathbf{p}_{goal}} \tag{10}$$

$\mathbf{V}_{in}, \mathbf{P}_{in} \in R^3$ are the velocity and position when the manipulator enter the velocity field.

After obtaining the velocity $\mathbf{V}$, the next position could be calculated combined with the current position and Δt (same with Δt in eq. (8)). From eq. (9), it can be found that with $(\mathbf{P} - \mathbf{P}_{goal})$ decreasing, $\mathbf{V}$ is reduced continuously to zero at the target position. Fig. 3 – Fig. 4 shows the effect of applying the method base on the velocity field.

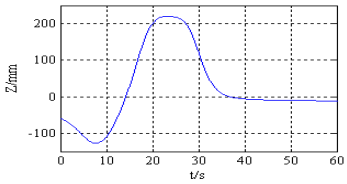

Fig. 3. The reproduced trajectory in Z-axis without the velocity filed method

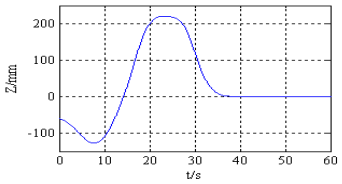

Fig. 4. The reproduced trajectory in Z-axis with the velocity filed method

Fig. 3 shows the result without the velocity field method, and it converges to -12mm finally. However, the result in Fig. 4 converges to 0 (the target position).

3 Experiments and Discussions

3.1 Experiments Setup

A series of experiments were conducted on the experimental platform, shown in Fig. 5, to verify above algorithms. The platform is mainly consisted of the 7-DOF manipulator, the central controller, the central control computer, the teleoperation computer, two typical obstacles and other subsystems such as the power system and the displayer.

The manipulator was taught to bypass the two obstacles starting from the left edge of the black rectangle area in Fig. 5, and terminate at the target position at the right. In this task, the diameter of the manipulator's end-effect was 120mm, and the center distance between the two obstacles was 350mm. One of the two obstacles was a cylinder with a diameter of 40mm, and the other one was a cuboid with the maximum upper surface of 80mm*40mm. Thus, in order to accomplish the task, the position of the manipulator had to be controlled precisely during the task (the orientation was

locked). The purpose of this experiment was to verify whether the method above could reproduce the trajectories satisfying the strict requirement of the position.

Human operator guided the manipulator to demonstrate the task. To facilitate the calculation, the coordinate system $\{x', y', z'\}$ (Fig. 5) was established over the target position. And there was a fixed positional relationship between the target position and the global coordinate system $\{x, y, z\}$. The target position was acquired by the camera.

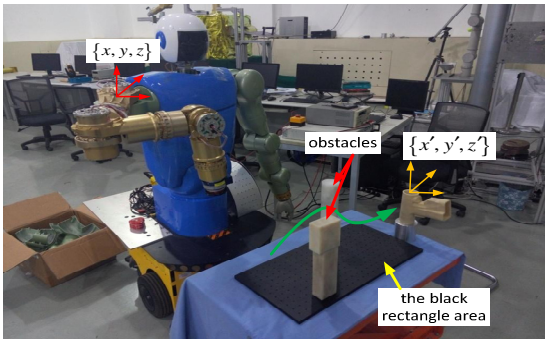

Fig. 5. The experiment platform

3.2 Result and Discussion

Aiming at reproduced trajectories, this paper presented three evaluation indexes: the convergence rate, the velocity characteristic, and the success rate.

The convergence rate con represented the rate of the reproduced trajectories which could converge to the target position. And it could be calculated as follows:

$$con = \frac{n_c}{N} \tag{11}$$

n_c expresses the number of the convergent trajectories. N indicates the total number of reproduced trajectories.

The velocity characteristic indicated the smoothness of velocity. And it limited the velocity was slower than 50mm/s and the acceleration was smaller than 20mm/s^2.

The success rate suc was the rate of completing the task. And it could be calculated as follows:

$$suc = \frac{n_s}{N} \tag{12}$$

N is the number of the reproduced trajectories which could complete the task.

As mentioned above, the three facts including the formulism methods of encoding the motions, the variables C in eq. (3) and σ in eq. (6) would affect the repro-

duced trajectories. In this section, taking advantage of the same demonstrations which is obtained by the experiments and changing the three facts referred above, we would compare and analyze the difference of results.

The Effect of the Formalism Methods

Different formalism methods meant different methods of encoding the motion, and would influence the information extracted from the demonstrations during training. Obviously, the formulism based on DS used the value of the sampling frequency of the sensors f_s and the single-step running time Δt while training and regressing. Regression results of the two formalism methods were shown in Fig. 6-Fig. 11:

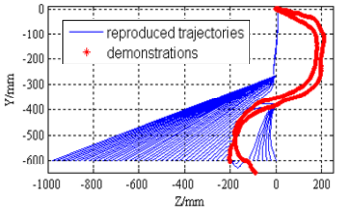

Fig. 6. The reproduced trajectories (based on positions)

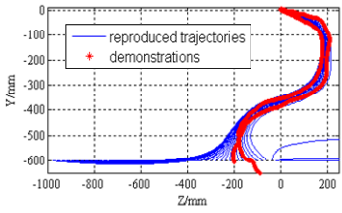

Fig. 7. The reproduced trajectories (based on DS)

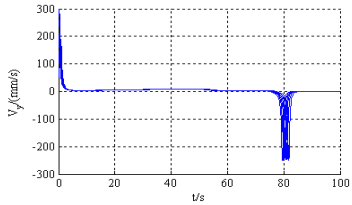

Fig. 8. The velocities of reproduced trajectories in Y-axis (based on positions)

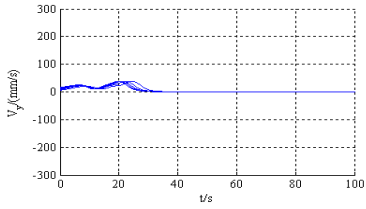

Fig. 9. The velocities of reproduced trajectories in Y-axis (based on DS)

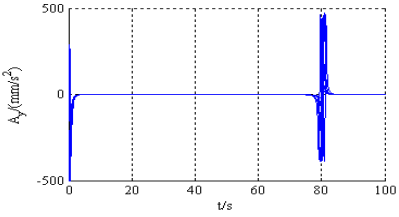

Fig. 10. The acceleration of reproduced trajectories in Y-axis (based on positions)

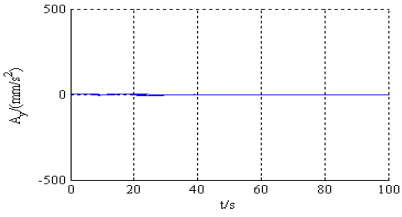

Fig. 11. The acceleration of reproduced trajectories in Y-axis (based on positions)

Because the left edge of the black rectangle area was at -600 in Y-axis of $\{x', y', z'\}$, the initial positions of the blue trajectories varied from (-1000, -600) to (0, -600), which took an interval of 20mm. During the execution of this task, the position changed very little (less than 10mm) in the X-axis. Therefore, the figures only showed the position in the Y-axis and Z-axis.

Comparing Fig. 6 with Fig. 7, it was found that the regression of the reproduced trajectories using the formalism based on positions (almost coincided with one of the demonstration trajectories) was better than those employing the formalism based on DS, if its initial position neared the demonstrations. But the reproduced trajectories using this formalism method had some drawbacks as follows: firstly, its generalization was bad ($con = 22\%$), and there would be local convergence, if the initial positions varied from (-1000, 600) to (-200, 60) as shown in Fig. 6. The success rate of that was only 20%. And then, from Fig. 8 and Fig. 10, it could be discovered that the velocity characteristic was bad. The maximum acceleration impact to 500mm/s^2, and the maximum velocity was faster than 250mm/s. Last but not least, this method depended on the teaching time excessively, and task execution time could not be adjusted in accordance with actual requirements.

In contrast, there were a lot of advantages of using the formulism based on DS. At first, its regression met the requirements of the obstacle avoidance. And it had a better generalization with success rate of 94%, as shown in Fig. 7. Then, in Fig. 9 and Fig. 11, the reproduced trajectories were smooth, and relied little on the teaching time.

The Influence of σ in Gaussian Kernel

The variable σ in eq. (6) can be considered to be the coverage of Gaussian kernel. In eq. (6), if the value of σ is large, the weight of high-dimensional feature will become very weak, and approximate to a low-dimensional subspace; on the contrary, arbitrary data would be linearly separable, and lead to the severe overfitting. Using the same demonstrations and formalism method, the trajectories were reproduced with the different values of σ (the value of the penalty function C was 16).

In Fig. 12-Fig. 14, the left and right showed the result with the same σ, and the differences between them were the restrictions such as the boundary, the obstacles (adding the compensation of the end-effector's diameter) and so on. If the value of σ was 300, as shown in Fig. 13, there were 47 reproduced trajectories of 50 ($suc = 94\%$) which had the good regression and convergence satisfying the task requirements by changing the initial position. When the value of σ was reduced to 210 (Fig. 12), two more reproduced trajectories could not converge to the target position ($con = 90\%$). And the reproduced trajectories were short of the regression if the initial position was from (-240, -600) to (-1000, -600), although they were more similar with the demonstrations if the initial position was nearby the demonstrations. Moreover, the success rate in Fig. 12 was only 26%. As the value of σ increased to 1166, all reproduced trajectories were convergent ($con = 100\%$) as shown in Fig. 14. But they had the bad regression and had the huge difference with demonstrations. Thus, 8% of them could meet the requirements ($suc = 8\%$).

As a result, the smaller the value of σ, and the worse success rate and convergence of the reproduced trajectories. And the larger the value of σ, the better the convergence

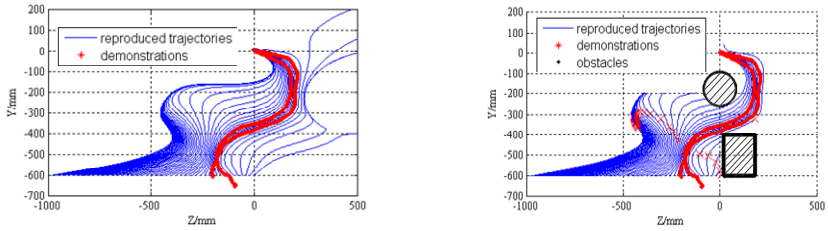

Fig. 12. The reproduced trajectories ($\sigma = 300$)

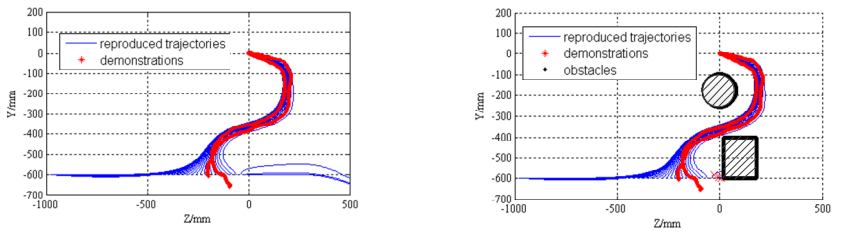

Fig. 13. The reproduced trajectories ($\sigma = 210$)

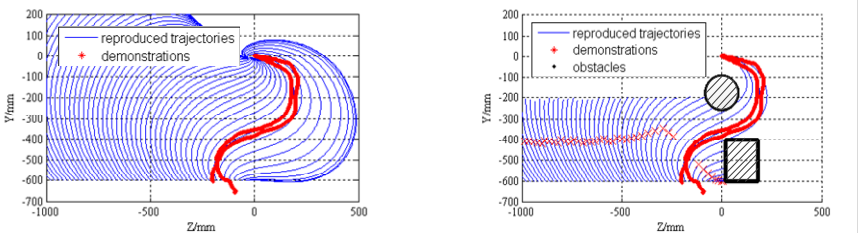

Fig. 14. The reproduced trajectories ($\sigma = 1166$)

and the worse success rate. So we should select the appropriate σ based on the actual system, the regression and convergence, to gain the best success rate.

The Effect of the Penalty Function C.

The main role of the penalty function C in eq. (3) is to balance the generalization of algorithm and the tolerance error. Fig. 15-Fig. 17 show the results with different value of C, using the same demonstrations and σ ($\sigma = 300$).

Comparing Fig. 15 and Fig. 16, the reproduced trajectories were dissimilar with the demonstrations if the value of C was smaller. At the same time, they were not good at the regression, and only 8 trajectories could ensure the task ($suc = 16\%$) in Fig. 15. And comparing Fig. 16 and Fig. 17, increasing value of C would lead to the regression deteriorate, though the results were more similar with the demonstrations as the initial position was near the demonstrations, and the success rate in Fig. 17 was 52%. However, it also

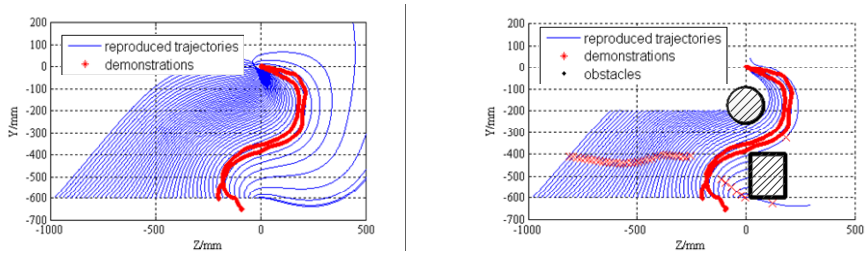

Fig. 15. The reproduced trajectories ($C = 0.2$)

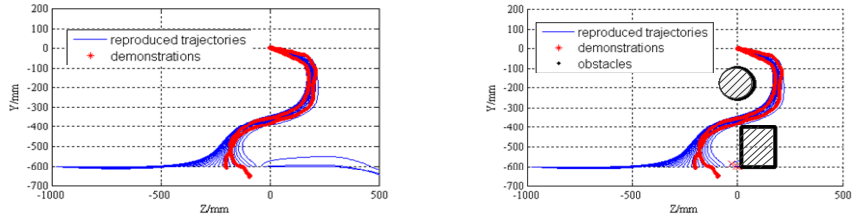

Fig. 16. The reproduced trajectories ($C = 16$)

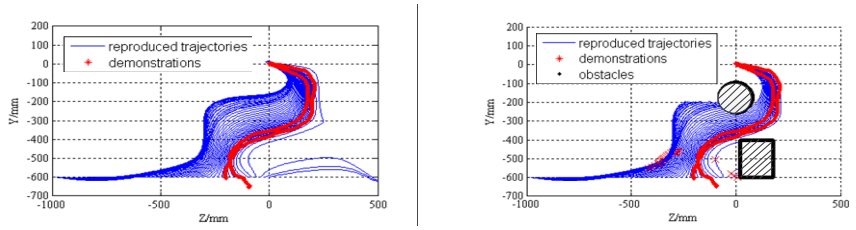

Fig. 17. The reproduced trajectories ($C = 50$)

could be concluded that the value of C had a little effect on the convergence of the trajectories ($con = 92\%$ in Fig. 15, $con = 94\%$ in Fig. 16 and Fig. 17).

4 Summary

In this paper, a new method of PbD was presented to meet the precise requirements of obstacle avoidance in tasks. Utilizing LS-SVM, this method has better generalization, and can obtain trajectories from wide-ranging initial positions with a few demonstration trajectories (only two in this paper). Furthermore, for obstacle avoidance and boundary constraints, the variables of σ and C can be adjusted according to the requirements of tasks. In addition, this method applies the velocity field method nearby the target position to maintain the convergence of the reproduce trajectories and smooth the velocity. In future work, we will make use of force sensors and visual sensors to monitor the real-time changes in the working environment of the manipulator, and exploit the online avoidance algorithm to improve the reproduced trajectories by PbD.

Acknowledgments. This work is in part supported by the National Natural Science Foundation of China (Grant No. 51305097), Natural Science Foundation of Heilongjiang Province(Grant No.E201428), the Fundamental Research Funds for the Central Universities(Grant No.HIT.KISTP.201410), and the Self-Planned Task(No.SKLR201404B) of State Key Laboratory of Robotics and System(HIT)

References

1. Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Robot programming by demonstration. In: Siciliano, B., Khatib, O., (eds.) Handbook of Robotics, ch. 59. Springer, New York (2008)
2. Schaal, S.: Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences **3**(6), 233–242 (1999)
3. Dong, S., Williams, B.: Learning and recognition of hybrid manipulation motions in variable environments using probabilistic flow tubes. International Journal of Social Robotics **4**(4), 357–368 (2012)
4. Tso, S.K., Liu, K.P.: Hidden Markov model for intelligent extraction of robot trajectory command from demonstrated trajectories. In: Proceedings of The IEEE International Conference on Industrial Technology (ICIT 1996), pp. 294–298. IEEE (1996)
5. Calinon, S., Guenter, F., Billard, A.: On learning, representing, and generalizing a task in a humanoid robot. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **37**(2), 286–298 (2007)
6. Calinon, S., Billard, A.: A probabilistic programming by demonstration framework handling constraints in joint space and task space. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008, pp. 367–372. IEEE (2008)
7. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives (2002)
8. Schaal, S.: Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In: Adaptive Motion of Animals and Machines, pp. 261–280. Springer, Tokyo (2006)
9. Ude, A., Atkeson, C.G., Riley, M.: Planning of joint trajectories for humanoid robots using B-spline wavelets. In: Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2000, vol. 3, pp. 2223–2228. IEEE (2000)
10. Gribovskaya, E., Billard, A.: Learning nonlinear multi-variate motion dynamics for real-time position and orientation control of robotic manipulators. In: 9th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2009, pp. 472–477. IEEE (2009)
11. Bentivegna, D.C., Atkeson, C.G.: Learning from observation using primitives. In: Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2001, vol. 2, pp. 1988–1993. IEEE (2001)
12. Khansari-Zadeh, S.M., Billard, A.: Learning stable nonlinear dynamical systems with gaussian mixture models. IEEE Transactions on Robotics **27**(5), 943–957 (2011)
13. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
14. Suykens, J.A.K., Vandewalle, J.: Least Square Support Vector Machine Classifiers. Neural Processing Letters **9**(3), 293–300 (1999)