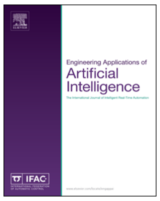
Engineering Applications of Artificial Intelligence 53 (2016) 32–40

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Robot trajectory generation using modified hidden Markov model and Lloyd's algorithm in joint space

Javier Garrido [a], Wen Yu [a,*], Xiaoou Li [b]

[a] Departamento de Control Automatico, CINVESTAV-IPN (National Polytechnic Institute), Mexico City 07360, Mexico
[b] Departamento de Computacion, CINVESTAV-IPN (National Polytechnic Institute), Mexico City 07360, Mexico

ARTICLE INFO

Article history:
Received 6 June 2014
Received in revised form
7 September 2015
Accepted 21 March 2016
Available online 11 April 2016

Keywords:
Hidden Markov model
Robot
Trajectory generation
Joint space

ABSTRACT

Human guide robots usually generate desired trajectories from human demonstrations. The training process can be in task space or joint space. The task space method needs the inverse kinematics; the joint space method uses dynamic time warping. Both of them destroy the accuracy of the generated trajectory.

In this paper, we first use Lloyd's algorithm to encode the input signals such that the observations are time-independent. The desired trajectory is generated in joint space without dynamic time warping. Then we modify the hidden Markov model (HMM) such that it can work in joint space. Since the desired trajectories are the joint angles, they can be applied directly to robot control without calculating the inverse kinematics. Simulation and experimental results show that the modified HMM with Lloyd's algorithm work well in joint space.

1. Introduction

The control structure of many human guide robots, such as robot exoskeleton, humanoid robot, and surgical robot, has three levels: joint angle tracking, trajectory planning, and path planning, see Fig. 1. The path planning is in task space. It generates a path from a starting-point to an ending-point with respect to some restrictions. The trajectory planning can be in task space or joint space. It gives the desired trajectories of the end-effector (task space), or desired joint angles (joint space), which are in the path generated by the path planning. The joint angle tracking usually uses PD/PID controller to force the joints to follow the desired angles generated by the trajectory planning.

The final problem of this control structure is to generate desired joint angles which satisfy human requirements (Xu and Lee, 2005). This is the object of transferring human skill to the robot through demonstrations. We also call it programming by demonstration (PbD) or learning from demonstration (LfD) (Argalla et al., 2009). The robot trajectory generation can be broadly divided into two trends: (1) Symbolic level. The human skill is decomposed into a sequence of action–perception units, then a statistical model is used to deal with the demonstrations (Hersch et al., 2008; Billard et al., 2007). (2) Trajectories level. A nonlinear mapping is used to model the sensor/motor information. The trajectories level method is robust to the environment changes (Ijspeert et al., 2012). The symbolic level method is suitable to model complex human actions. In this paper we use the symbolic level method.

Although the trajectory planning in joint space can avoid the calculation of the inverse kinematics, the demonstrations in joint space are time-dependent. Fig. 2 shows the trajectories in joint space and task space, when a two-link planar robot draws a broken line. Since the trajectories in task space only give space relation, the three lines overlap in task space. However, in joint space they are complete different because of different drawing speeds. In this sense, training in task space is easier than in joint space (Vakanski et al., 2012; Kuniyoshia et al., 2004; Wang et al., 2013; Gribovskaya et al., 2011). After the training in task space, the inverse kinematics need to be solved, which requires complete knowledge of the robot.

There are few works in joint space (Berg et al., 2010; Peters and Schaal, 2008). The dynamic time warping (DTW) is an effective tool to deal with the time-dependent problem. The computation time for one-dimensional signals, such as time series, is in polynomial. The extension of DTW for more than two-dimension, like robots, becomes NP-complete. The accuracy of the high dimension approximation is also very low (Sakoe and Chiba, 1978).

Statistical learning techniques deal with the high variability inherent in the demonstrations. They are not sensitive to disturbances. For instance, spline smoothing technique can deal with the uncertainty in several motion demonstrations (Ude, 1993). The mean and variance of the collected variables are applied in Ogawara et al. (2003) to generate a model. Ito et al. (2006) realizes online imitation by encoding two different motor loops.

Hidden Markov Model (HMM) generates a sequence which is

* Corresponding author.
E-mail address: yuw@ctrl.cinvestav.mx (W. Yu).

http://dx.doi.org/10.1016/j.engappai.2016.03.006
0952-1976/© 2016 Elsevier Ltd. All rights reserved.

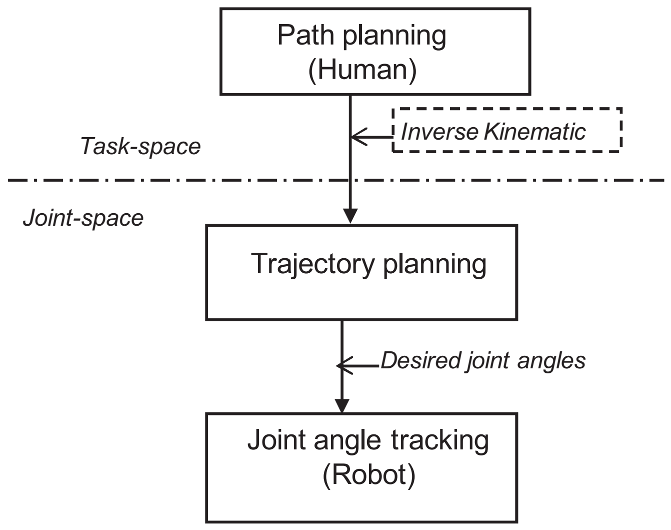


Fig. 1. Control structure of a human guide robot.

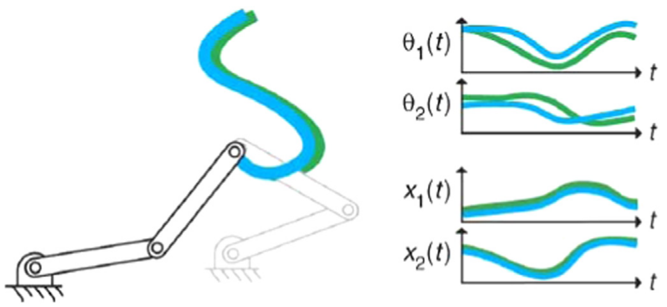


Fig. 2. A two-link planar robot draws a broken in joint space and task space.

called Markov chain (Rabiner, 1989; Yang et al., 1997). It can encode the motion of a robot, and find the highest probability state path by Viterbi algorithm (Viterbi, 1967). HMMs use finite Gaussian mixture models as their hidden state distributions. The Gaussian mixture model can encode a set of trajectories (Argalla et al., 2009). The Gaussian mixture regression can retrieve a smooth trajectory from several demonstrations (Billard et al., 2007). There are many successful application of robot trajectory generation via HMM (Yang et al., 1997; Kwon and Park, 2008). HMM offers many advantages over the other statistical models for human behavior modeling, such as better compression, variant structures, and training incrementally. One weakness of HMM is that the trajectory generation can only use current state for the emission and the transition probabilities. HMM does not map well to many time-dependent domains, such as joint space (Mohamed and Gader, 2000).

To train HMM, it is necessary to map continuous trajectories into discrete values, named codebook. It is impossible to use all sampled data to train HMM. The key-points include necessary information for HMM. The normal method of selecting the key-points uses the shape of the trajectory. It can be positions evaluation (Yang et al., 1997) or position/velocity evaluation (Vakanski et al., 2012). Linde–Buzo–Gray (LBG) is the most popular method. The above methods do not work well in joint space, because the trajectories in joint space are time-dependent, while these methods use the shape information (Gernot, 2008). Lloyd's algorithm partitions data into well-shaped and uniformly sized convex cells (Lloyd, 1982). It repeatedly finds the centroid of each set in the partition using Voronoi diagrams. In this paper, we use Lloyd's algorithm to avoid the following two problems: (1) calculation problem of the dynamic time warping in joint space, (2) selection problems of the codebook and key-points in joint space. The classical HMM is also modified such that it is more feasible to generate joint trajectory. A new element is introduced for HMM.

In order to generate the desired robot trajectory, this paper proposes a different method from the previous works like inverse kinematics and dynamic time warping. The contributions of the paper include three parts:

1. The classical HMM is modified such that it is more feasible to generate trajectory in joint space. We introduce a new auxiliary output in HMM to help training process.
2. The Lloyd's algorithm is modified for HMM, such that it can solve the problems in joint space learning.
3. The proposed method is validated with a two-link planar robot and a four degree-of-freedom (4-DoF) exoskeleton robot (Garrido et al., 2016).

2. Codebook and key-points generation

The dynamics of a n–link robot in joint space can be written as

$$M(q)\ddot{q} + c(q,\dot{q})\dot{q} + g(q) + F(\dot{q}) = \tau \tag{1}$$

where $q \in \Re^n$ denotes the links angles, $\dot{q} \in \Re^n$ denotes the links velocity, $M(q) \in \Re^{n\times n}$ is the inertia matrix, $c(q,\dot{q}) \in \Re^{n\times n}$ is the centripetal and Coriolis matrix, $g(q) \in \Re^n$ is the gravity vector, $F \in \Re^n$ is the frictional terms, and $\tau \in \Re^n$ is the input control vector.

The object of this paper is to generate the desired trajectory q^* from the demonstrations in the joint space $Q = [Q_1 \cdots Q_n]^T$, $Q_i = [X_i^1 \cdots X_i^m]^T$. Here the trajectory X_i^r defines i-th joint and r-th trajectory, it is defined as $X_i^r = \{q_i(l)\}$, $l = 1 \cdots T_i^r$, T_i^r is the total sample number of the joint angles. There are m trajectories for each joint. The trajectory number m is not necessary large, because human cannot repeat the same motion so many times and this may cause repeat calculation in HMM. The data length T_i^r is different from one demonstration to another, because of difference speed, and start/ending time.

Each joint for each demonstration has its own codebook defined as C_i, $i = 1 \ldots n$. A codebook can be regarded as an N_i dimension vector, i.e., $C_i = [c_{i1} \cdots c_{iN_i}]^T$. The codebook dimension N_i is selected by prior knowledge of the trajectories geometry.

In this paper we use Lloyd's algorithm to train the codebook $C_i(t)$, here t is the training times. The initial codebook is defined as $C_i(1)$. A bad initial condition may reach local minima. A heuristics method (Zafra and Pechenizkiy, 2013) is used to find a better $C_i(1)$. Since the heuristics method needs a lot time, $C_i(1)$ can also be selected randomly from X_i^r.

The objective of using Lloyd's algorithm to create the codebook is to minimize a quantization error with certain data distribution. We need nearest-neighbor and centroid conditions which are commonly used in Lloyd's algorithm. For one point $q_i(l_1)$ in the trajectory, we want to find the nearest codebook element by calculating

$$\min_{1\le j\le N} |q_i(l_1) - c_{ij}|, \quad l_1 = 1 \cdots T, \quad i = 1 \cdots n \tag{2}$$

If the nearest codebook element is c_{ik}, the region R_{ij} $(i = 1 \cdots n,\ j = 1 \cdots N_i)$ is defined as

$$R_{ik} = \{q_i(l_1)\} = [r_{ik1} \cdots r_{ikp_{ik}}], \quad k = 1 \cdots s_i^r \tag{3}$$

where p_{ik} is the length of the region R_{ik}, s_i^r is the region number of the joint i and the demonstration r.

Obviously, the center of the region R_{ij} is should be c_{ij}. The new center of R_{ij} can be calculated as