2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Generating Manipulation Trajectory Using Motion Harmonics

Yongqiang Huang and Yu Sun

Abstract— **This paper presents a novel manipulation trajectory generating algorithm that constructs trajectories from learned motion harmonics and user defined constraints. The algorithm uses functional eigenanalysis to learn motion harmonics from demonstrated motions and then use the motion harmonics to compute the optimal trajectory that resembles the demonstrated motions and also satisfies the constraints. The algorithm has been tested on five real human motion data sets to obtain motion harmonics and then generate motions of each task for a NAO robot. The generated trajectories were compared with the trajectories generated using linear segment with parabolic blend approach and with the Open Motion Planning Library. The approach can also work with motion planners.**

I. INTRODUCTION

Learning from demonstration (LfD) [1] is a widely used technique that is often used to teach robot motions. The goal of teaching a robot is not to replicate a task, but rather, to convey the essence of the task, so that the task can be undertaken in a new environment with possibly different constraints. Since the new/different constraint has not been taught, by completing the task while satisfying that constraint, the robot essentially generates a new motion. In the graphics community, this problem is often referred to as motion synthesis, where the robots are replaced with animated characters.

Extensive research has been conducted related to motion generation/synthesis, and a number of solutions have been proposed. As a valid model for time representation, the hidden Markov model (HMM) has been successful in human motion modeling. For example, it is used as the second level of the two-level model of motion data in [2]. In [3], a multidimensional HMM is used to encrypt the styles of motions, and to generate new styles. Similar to [3], also focusing on variation/styles of motion, [4] learns the structure of a dynamic Bayesian network, which has the ability to synthesize both temporal and spatial variants of the original motion.

The linear dynamical systems (LDS) provides another solution. [5] uses LDS for modeling motion textons and transition matrices for texton distribution, and synthesizes motion sequences with constrained LDS, while [6] presents motions with LDS, and generates new motions using a optimization technique which considers both Gaussian-modeled motion priors and user defined constraints. Different from LDS, [7] uses Gaussian process (GP) to model the force field that exists in the motion, and combines it with a Newtonian dynamics model to synthesize new motions. In [8], GP is used to model transitions between morphable primitives, which is a model that encrypts geometric and time variation. In [9], the motion is modeled as a directed graph where the edges represent clips of motion and the nodes represent their connections. The branch and bound algorithm is used to search for a path that meets the user's requirements. [10] builds a hierarchical graph to represent motion sequences, and uses random search to look for paths that accommodates to user constraints. On the basis of [9], [10], and [11], [12] builds a compressed interpolated motion graph, and uses ARA*, an anytime heuristic search algorithm to find the optimal or sub-optimal path in the graph that meets the path sketched by the user and also compute the interpolation weights.

Besides the above mentioned models, the movement primitive is another approach for motion modeling [13]. In the movement primitive framework, a discrete motion is represented by an attractor dynamical system and a cyclic motion is represented by a limit-cycle dynamical system. The framework uses two sets of dynamical systems: one for dynamical behavior, the other for following a desired path. Principal component analysis (PCA) is used in [14] to process the motion. New motions are obtained through optimization with constraints on energy and trajectory smoothness. [15] considers each motion time series as a high-dimensional vector, and generates trajectories by interpolating the principle components obtained through PCA. In [8] and [16], PCA is used for analyzing the geometric variation and timing variation separately, and the results are combined to form a single deformable model. The new motions are obtained through the maximum a posteriori framework.

Another way of dealing with time series is functional data analysis, which treats data as discrete observations from continuous functions [17]. Different from vectors, which represent time by the order of points, and from HMM, which represent time using states, functions embed time in a continuous manner and imitates actual motions more naturally, and therefore has become a powerful tool in motion analysis. [18] applies functional PCA for arm gesture recognition and whose efficiency exceeds that of dynamic time warping (DTW). [19] combines PCA and functional PCA to analyze human grasp types and arrives at a new taxonomy of grasp types. Besides computer vision and robotics, functional PCA has found use in other fields such as brain science [20]. The application of functional analysis on motion generation, however, has been rare.

This paper contributes a unique framework of motion generation. Our approach applies functional analysis to ex-

The authors are with the Department of Computer Science and Engineering at the University of South Florida, Tampa, FL, USA. `yongqiang@mail.usf.edu`, `yusun@cse.usf.edu`

tract motion harmonics from demonstrated motion, takes constraints from the user, and generates new motions that balance meeting user constraints with resembling learning data through optimization. The approach can potentially work with motion planners to clear obstacles in the joint space.

The paper first gives a concise overview of the approach in section II-A. Then, it covers data preprocessing in section II-B, explains how the motion harmonics are extracted in section II-C, and how they are used to construct new motions together with constraints in section II-D and II-E. Section II-F and II-G define the dissimilarity and the error measure used for evaluation. In the end, the proposed approach is evaluated in experiments from three different aspects and the results are presented and discussed in section III.

II. Approach

A. Approach overview

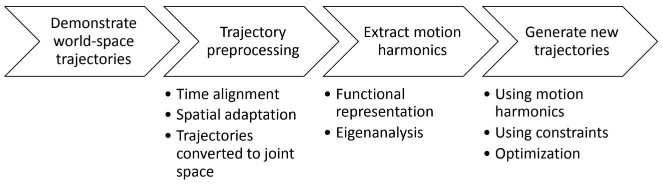

Fig. 1. The proposed approach takes the demonstrated trajectories and goes through three major steps. First, the data go through temporal and spatial preprocessing and switch to a different space, and motion harmonics are extracted from the data. Then the approach takes task constraints, and uses the motion harmonics to generate new motion through optimization.

Fig. 1 shows the procedures of the proposed motion generation approach. First, human motion trajectories in the world space are collected. Since the trajectories may not lie completely within the workspace of the robot, we adapt them and meanwhile perform inverse kinematics to convert them to the robot's configuration space or joint space. Instead of discrete vectors, we use continues functions to represent the trajectories. Based on the data's functional representation, a series of continues motion harmonics are obtained through functional analysis. To generate a new trajectory for a task with novel constraints that could be related to a new environment or a new goal, an optimal composition of the motion harmonics is computed to find a trajectory with the goal of resembling the demonstrated trajectories and minimizing the distance between the trajectory and the given constraints. The generated trajectory is in the robot's joint space and can be directly used as joint control inputs. The algorithm in this paper is presented to work in the joint space because of our intention of facilitating joint control. It would work in the world space equally well.

B. Motion Data Preprocessing

We preprocess the data before extracting the motion harmonics. First, we align all trajectories in time using batch DTW [21]. If the robot's workspace is smaller than demonstrated workspace, the trajectories are then adapted to fit in the robot's workspace by iterative downscaling and translation. The adaptation returns the final scale and translation of the entire set of trajectories, and the inverse kinematics of each single trajectory.

C. Extracting motion harmonics

1) Representing trajectories with functions: In the real world, physical quantities change continuously in time. In contrast, most data people collect are discrete in time due to the limited sampling rate of measuring devices. Coming from physical quantities, data are intrinsically continuous and therefore should be treated accordingly. Let $\mathbf{x} = [x_1, x_2, ..., x_T]^T \in \mathbb{R}^{T\times 1}$ denote some human motion data collected uniformly in time where T is the total number of samples. Using $\mathbf{x}$ directly as a T-vector in analysis fails to preserve the very essential characteristics of human motion which is continuous. To remedy that, we consider $\mathbf{x}$ as being driven by a function $x(t)$, and $\mathbf{x}$ as discrete samples of $x(t)$ collected along axis t with measurement noise.

A general function that does not have an explicit expression can be expressed using a basis expansion. Denote by $\{\phi_k(t)\}$, $k = 1, ..., K$, a general functional basis system, and $\{c_k\}$, the corresponding coefficients. A general function $f(t)$ can be expressed as

$$f(t) = \sum_{k=1}^{K} c_k \phi_k(t). \tag{1}$$

To represent $\mathbf{x}$ using a basis expansion as in Eq. (1), both $\{\phi_k(t)\}$ and $\{c_k\}$ must be determined.

$\{\phi_k(t)\}$ should be determined according to the characteristics of data $\mathbf{x}$. For open-ended non-periodic data, the spline basis offers modeling flexibility through the choice of order and design of breakpoints. For data that exhibits periodic patterns, the Fourier basis is a natural candidate.

With a chosen basis system $\{\phi_k(t)\}$, c_k's are computed by fitting the basis $\{\phi_k(t)\}$ to the data $\mathbf{x}$. Since $\phi_k(t)$ is defined within time interval $[1, T]$, the basis $\phi_k(t)$ can be sampled as $\boldsymbol{\phi}_k = [\phi_{k,1}, \phi_{k,2}, ..., \phi_{k,T}]^T \in \mathbb{R}^{T\times 1}$ within $[1, T]$. We define $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_K]$ and $\mathbf{c} = [c_1, c_2, ..., c_K]^T$, then the data $\hat{\mathbf{x}} \in \mathbb{R}^{T\times 1}$ can be proximated by

$$\hat{\mathbf{x}} = \boldsymbol{\Phi}\hat{\mathbf{c}}, \tag{2}$$

where

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} (\mathbf{x} - \boldsymbol{\Phi}\mathbf{c})^T \mathbf{W} (\mathbf{x} - \boldsymbol{\Phi}\mathbf{c}), \tag{3}$$

and $\mathbf{W} \in \mathbb{R}^{T\times T}$ is a symmetric weighting matrix that accounts for non-uniform variance along time range $[1, T]$.

$\hat{\mathbf{c}}$ specifies the approximated driving function $\hat{x}(t) = \sum_k \hat{c}_k \phi_k(t)$. Particularly, we may want $\hat{x}(t)$ to be smooth. We can define a smoothness penalty

$$P_{\text{smooth}} = \int_1^T [\hat{x}''(t)]^2 dt = \int_1^T \left[\sum_{k=1}^{K} \hat{c}_k \phi_k''(t)\right]^2 dt. \tag{4}$$

Let $D^2\boldsymbol{\Phi}$ be $\{\phi_k''(t)\}$ sampled along $[1, T]$. Thus,

$$P_{\text{smooth}} = ||(D^2\boldsymbol{\Phi})\mathbf{c}||^2. \tag{5}$$

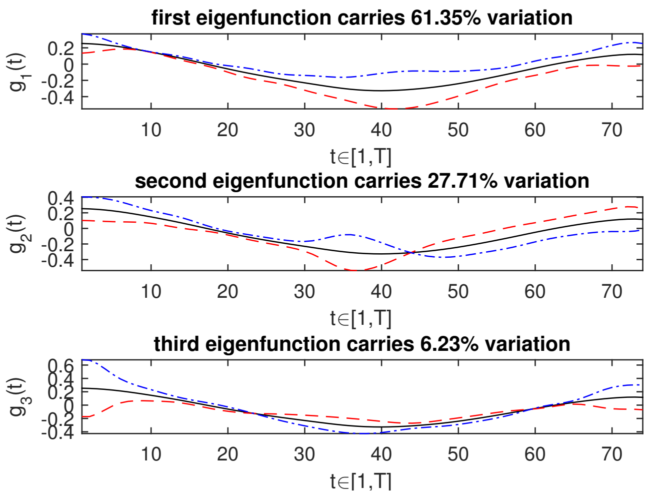

Fig. 2. Illustration of different eigenfunctions carrying different variation. In each subfigure, the mean function is shown as black solid line, the mean function plus the eigenfunction is shown as blue dash-dotted line, and the mean function minus the eigenfunction is shown as red dashed line.

$\hat{\mathbf{c}}$ is found by

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} \left[(\mathbf{x} - \mathbf{\Phi c})^T \mathbf{W} (\mathbf{x} - \mathbf{\Phi c}) + \lambda P_{\text{smooth}} \right], \quad (6)$$

where λ weighs P_{smooth} against $(\mathbf{x} - \mathbf{\Phi c})^T \mathbf{W} (\mathbf{x} - \mathbf{\Phi c})$.

2) Functional analysis: After adaptation, the demonstrated motion trajectories are converted into the joint space. Consider a set of joint space trajectories represented by functions: $q_i(t)$, where $i = 1, \ldots, N$, and $t \in [1, T]$. N is the number of trajectories.

The motion harmonics are the eigenfunctions of $\{q_i(t)\}$. We explain the acquisition of motion harmonics using the simplest case where $q_i(t)$ is one dimensional. To get the motion harmonics, we first calculate the mean motion

$$q_0(t) = \frac{1}{N} \sum_{i=1}^{N} q_i(t), \quad (7)$$

and use it to center all the trajectories:

$$q_i^*(t) = q_i(t) - q_0(t). \quad (8)$$

The covariance function is defined as

$$v(t,s) = \frac{1}{N} \sum_{i=1}^{N} q_i^*(t) q_i^*(s), \quad (9)$$

and the eigenfunctions $g(t)$ are determined by solving

$$\int_1^T v(t,s) g(s) ds = \lambda g(t), \quad (10)$$

where λ is the eigenvalue corresponding to $g(t)$. Different eigenfunctions carry different variation in the data, and a simple example is shown in Fig. 2.

We select the M eigenfunctions $g_m(t)$, $m = 1, .., M$, with the largest eigenvalues, and refer to them as motion harmonics.

D. Constructing trajectories using motion harmonics

Using M eigenfunctions $g_m(t)$, $m = 1, ..., M$, a new trajectory can be constructed by

$$q(t) = q_0(t) + \sum_{m=1}^{M} c_m g_m(t), \quad (11)$$

where c_m is the coefficient of $g_m(t)$. Since $\{g_m(t)\}$ comes from the data, by using them one can only generate trajectories that lie within the range of variation in the data. To allow shifting of new trajectories, we extend $\{g_m(t)\}$ and add a constant basis $g_{M+1}(t) = 1$:

$$\{g_m(t)\}_{m=1}^{M'} = \{g_m(t)\}_{m=1}^{M} \cup \{g_{M+1}(t)\}, \quad (12)$$

where $M' = M + 1$. Thus, a new trajectory is constructed by

$$q(t) = q_0(t) + \sum_{m=1}^{M'} c_m g_m(t). \quad (13)$$

The construction of a new trajectory is determined by the coefficient $\{c_m\}$, $m = 1, .., M'$.

E. Incorporate Constraints

1) Timed configuration constraints: For the learned task to be performed in a novel environment or for new goals, a set of N_c constraints can be specified and are denoted as $\{e_i\}$, $i = 1, 2, \ldots, N_c$. The constraints specify the joint-space configuration at time instants $\{t_i\} \in [1, T]$, $i = 1, 2, \ldots, N_c$.

The joint space trajectories $\{q_i(t)\}$, $i = 1, .., N$, can be approximated using the motion harmonics $\{g_m(t)\}_{m=1}^{M}$ (i.e., without the constant basis $g_{M+1}(t) = 1$), with certain coefficients $\{c_{m,i}\}$. The coefficients are obtained by computing

$$c_{m,i} = \int_1^T (q_i(t) - q_0(t)) g_m(t) dt, \quad (14)$$

whose mean is

$$\bar{c}_m = \frac{1}{N} \sum_{i=1}^{N} c_{m,i} \quad m = 1, ..M. \quad (15)$$

We want to construct a new trajectory in a way that takes the constraints into consideration and also respects the demonstrated trajectories. The new trajectories is constructed by solving

$$\min_{c_m} \quad \frac{1}{N_c} \sum_{i=1}^{N_c} [e_i - q_0(t_i) - \sum_{m=1}^{M'} c_m g_m(t_i)]^2 + \frac{\alpha}{M} \sum_{m=1}^{M} (c_m - \bar{c}_m)^2, \quad (16)$$

where α is the weighting factor that balances between making the new trajectory stay close to the demonstration and making it meet the constraints.

2) Joint angle limits: Since $q(t)$ must lie within the joint space so that the robot can physically perform it, we add the constraints of joint angle range

$$q_l \leq q_0(t) + \sum_{m=1}^{M} c_m g_m(t) \leq q_u, \quad t \in [1, T] \tag{17}$$

where q_l and q_u are the lower and upper bound, respectively, for the joint.

The problem posed by Eq. (16) and (17) can be solved by quadratic programming.

F. Dissimilarity measure

A dissimilarity measure is defined for the evaluation of our motion generation approach. It measures how dissimilar a trajectory is to a set of other trajectories.

A newly generated world-space trajectory y is compared with every non-preprocessed demonstrated world-space trajectory x_i, $i = 1, .., N$, where y and x_i are both discretely sampled time series. We use DTW to quantify the distance between y and x_i because DTW is a main benchmark of similarity measures for time series and very few similarity measures have been reported to systematically outperform DTW [22], [23]. However, since DTW is distance based, the accumulated distance matrix generated using two trajectories that are far away and with different scales may lead to incorrect alignment or incorrect normalized distance between the trajectories. To avoid that potential problem, we scale and translate each x_i before comparing it with y, using the final scale s_{final} and final translation d_{final} returned by adaptation. Let trajectory x_i be specifically expressed as a time series $x_i = \{x_{i,1}, x_{i,2}, ..., x_{i,T}\}$, where T is the number of samples. The center of x_i is $\bar{x}_i = \sum_{t=1}^{T} x_{i,t}$, and x_i that is scaled and translated is

$$x_i^* = (s_{\text{final}}(x - \bar{x}_i) + \bar{x}_i) + d_{\text{final}}. \tag{18}$$

We define the distance between x_i^* and y as the normalized minimum distance between x_i^* and y computed by DTW, and denote it as $DTW(x_i^*, y)$. The dissimilarity between y and $\{x_i\}$ is the average distance between y and $\{x_i^*\}$. We assume the dissimilarity is always measured between the data $\{x_i\}$ and the new trajectory y that uses the data, and therefore we omit $\{x_i\}$ when we talk about dissimilarity and only mention y:

$$\text{dissimilarity}(y) = \frac{1}{N} \sum_{i=1}^{N} DTW(x_i^*, y). \tag{19}$$

G. Error measure

We measure how well trajectories generated by our approach meet the timed constraints by defining the average world-space error:

$$\text{error}(y) = \frac{1}{NN_c} \sum_{i=1}^{N} \sum_{j=1}^{N_c} |y(t_j) - f(e_j)|, \tag{20}$$

where y is the generated trajectory and e_j is the j-th constraint at time t_j.

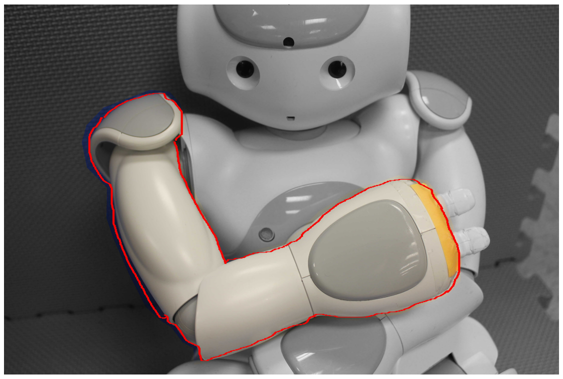

Fig. 3. We define the right arm of NAO as the kinematics chain.

III. EXPERIMENT

A. Preparing data

1) Obtaining data: We tested our approach using five sets of data taken from [24]: *beat*, *hand over*, *answer phone*, *pull gun*, and *punch*. Among those tasks, *answer phone* and *pull gun* are performed by the same person, and each of the rest three is performed by a different person. Each task is repeated a number of times, and there are a total of 63 repetitions/trials.

The robot on which we tested our approach is NAO H25 v3.3. We set the hand as the end effector and consider the right arm as the kinematics chain (Fig. 3). Thus, the root joint for human is *R_collar* [24] and for NAO it is *RShoulderPitch*. We include three more joints for NAO: *RshoulderRoll*, *RElbowYaw*, and *RElbowRoll*. Thus our robot model has a degree of freedom (DoF) of four, which severely limits the options of orientation when it reaches certain positions. Hence, in this paper, we only use the position information (x, y, z), and thus, each motion trial is a three-dimensional trajectory. The distance in the world space is measured in millimeter.

2) Preprocessing: Our implementation of DTW uses the Sakoe-Chiba local constraint with a slope range of $[0.5, 2]$ and which implies the Itakura global constraint.

Being an iterative process, the adaptation is affected by the initial scale and inter-iteration scaling factor. A large initial scale and a small scaling factor makes the algorithm runs slower, but gives it higher probability to converge.

B. Motion generation

Ramsay's FDA Matlab package [17] is used to represent trajectories with functions and obtain motion harmonics. The eigenanalysis considers all the joints together. Fig. 4 shows the first three eigenfunctions for dataset *beat*. Twenty B-spline basis functions are used for functional representation. The implementation of quadratic programming is from MathWorks [25].

C. Evaluation

We evaluate our approach by comparing it with two other trajectory generating approaches and by testing our approach on avoiding obstacles with the guidance of via points.

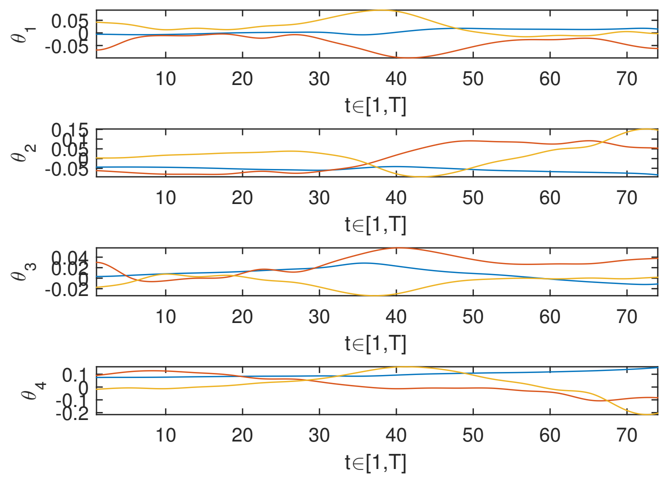

Fig. 4. First three eigenfunctions of dataset *beat* for each degree

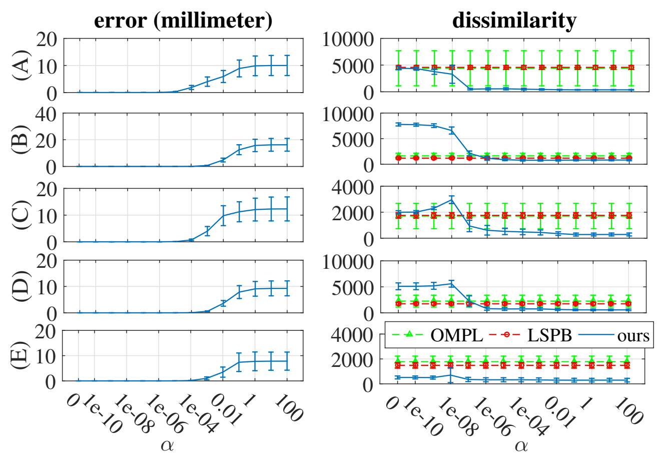

Fig. 5. The left column shows the error of our method, and the right column shows the dissimilarity of OMPL, LSPB, and our method. Each row corresponds to one dataset: (A) beat (B) handover (C) answer phone (D) pull gun and (E) punch.

1) *Comparing with Linear Segment Parabolic Blend:* First, we compare with the classical *Linear Segment with Parabolic Blend* approach (LSPB) [26]. Let $\{x_i\}$ be the world-space trajectories returned by adaptation, where each $x_i = \{x_{i,1}, x_{i,2}, ..., x_{i,T}\}$ is a time series. The mean trajectory $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_T\}$. We select the start point $x_s = \bar{x}_1 + 20 \cdot [r_1, r_2, r_3]^T$, and the end point $x_e = \bar{x}_T + 20 \cdot [r_1, r_2, r_3]^T$, where $r_1, r_2, r_3 \sim U(0, 1)$. Then we compute their configuration $q_s = f^{-1}(x_s)$ and $q_e = f^{-1}(x_e)$, where $f^{-1}()$ represents inverse kinematics. We specify two timed configuration constraints: $\{e_i\} = \{q_s, q_e\}$ with time $\{t_i\} = \{1, T\}$. Our approach produces different trajectories given different weighting factor α. We choose 14 values of α from $[0, 100]$, and for each α, we run the approach 20 times. Thus, there are $14 \times 20 = 280$ different sets of constraints.

2) *Comparing with the Open Motion Planning Library:* Second, we compare with trajectories generated by control-based planners through the Open Motion Planning Library (OMPL) [27].

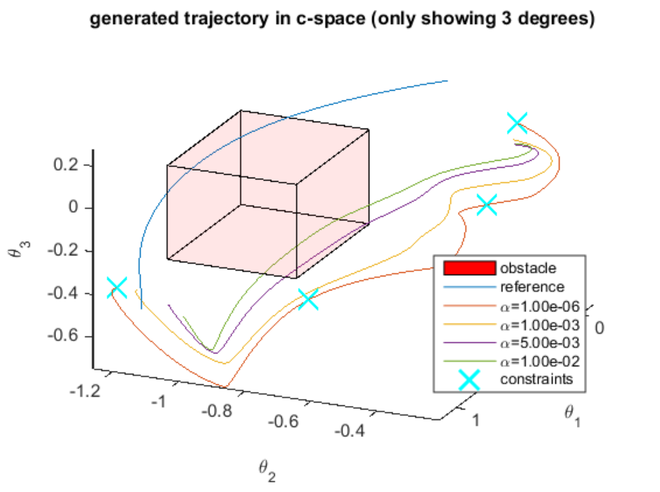

Fig. 6. result of experiment *clearing obstacle* with sufficient guidance from via points

The left column of Fig. 5 shows the average world-space error of constraints $\{q_s, q_e\}$. As α increases, the generated trajectory leans towards the demonstrated data and cares less about the constraints, and the average error increases. Conversely, as shown in the right column of Fig. 5, the dissimilarity of the generated trajectory goes down as α increases. When α reaches a certain point, the dissimilarity of our trajectories becomes lower than the dissimilarity of both the LSPB and the OMPL trajectories, and stays low thereafter. In addition, a range of α is observed for which both the average error and the dissimilarity are low. The αs in such range may be considered desirable for automatic motion generation.

The paper is accompanied by a video that shows the NAO robot executing trajectories of the experimented tasks generated by the compared approaches.

3) *Avoiding obstacles with guidance of via points:* Third, in addition to start and end points, via points are added to guide the trajectory. We test if the trajectory can clear an obstacle in the configuration space. The start and end points are inherited from the last two experiments, and the via points are the optimal path states generated by OMPL which clear the obstacle.

Fig. 6 shows sufficient guidance provided by the via points. When α is small, the trajectory weighs the via points more and by which avoids the obstacles. As α becomes larger, the demonstrated data shows more influence, and the trajectory hits the obstacle. Fig. 7 shows similar phenomenon. In contrast, when the guidance provided by the via points is poor, the trajectory hits the obstacle even if it strictly adheres to the via points. This is shown in Fig. 8, where the via point resides too close to the obstacle.

Currently, our approach does not specifically deal with obstacles, but as the results show, it can generate obstacle-clearing trajectories if quality via points are provided from motion planners. This preliminary result shows the ability of our approach to work with motion planners.

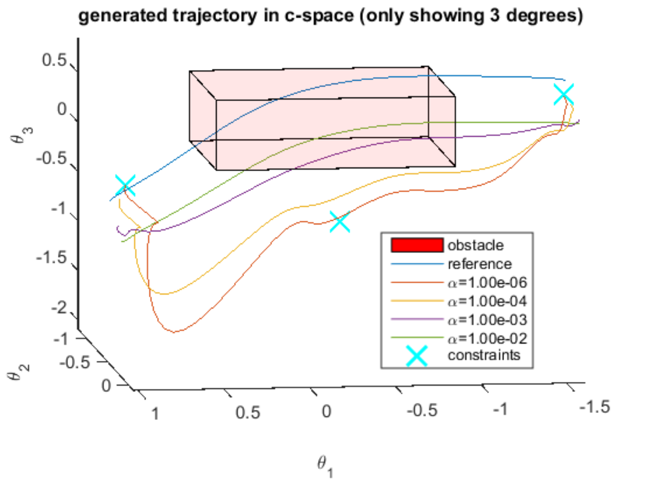

Fig. 7. result of experiment *clearing obstacle* with sufficient guidance from via points, second example

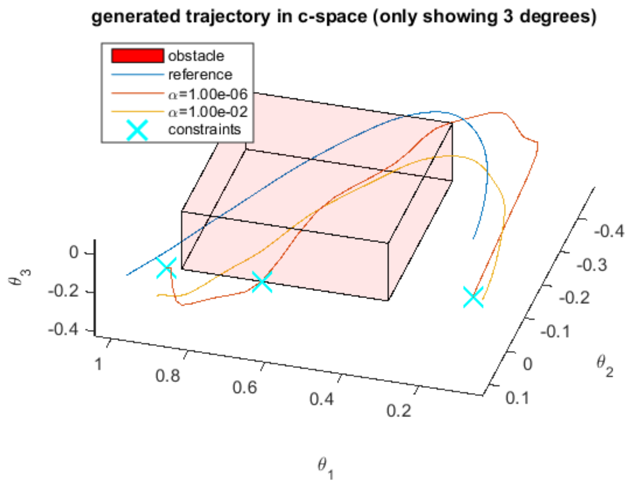

Fig. 8. result of experiment *clearing obstacle* with poor guidance from via points

IV. Conclusion

The paper presents a unique motion generation approach that uses functional analysis of motion and optimization to consider both demonstrated motion and user constraints. The trajectories generated by the approach have been compared with those generated by LSPB and OMPL to demonstrate the advantage of the balancing mechanism of the approach. With quality guidance points provided by a motion planner, the approach can clear obstacles in the joint space.

V. Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1421418.

References

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, “Robot programming by demonstration,” in *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.
[2] L. M. Tanco and A. Hilton, “Realistic synthesis of novel human movement from a database of motion capture examples,” in *Proceedings of the Workshop on Human Motion (HUMO’00)*, 2002, pp. 137–142.
[3] M. Brand and A. Hertzmann, “Style machines,” in *Proceeding SIGGRAPH ’00*, 2000.
[4] M. Lau, A. Bar-Joseph, and J. Kuffner, “Modeling spatial and temporal variation in motion data,” in *Proceeding SIGGRAPH Asia ’09*, 2009.
[5] Y. Li, T. Wang, and H.-Y. Shum, “Motion texture: A two-level statistical model for character motion synthesis,” in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, 2002.
[6] J. Chai and J. K. Hodgins, “Constraint-based motion optimization using a statistical dynamical model,” *ACM Trans. Graph*, 2007.
[7] X. Wei, J. Min, and J. Chai, “Physically Valid Statistical Models for Human Motion Generation,” *ACM Transaction on Graphics*, vol. 30, no. 19, 2011.
[8] J. Min and J. Chai, “Motion Graphs++: A Compact Generative Model for Semantic Motion Analysis and Synthesis,” *ACM Transaction on Graphics*, vol. 31, no. 153, November 2012.
[9] L. Kovar, M. Gleicher, and F. Pighin, “Motion graphs,” in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, 2002.
[10] O. Arikan and D. A. Forsyth, “Interactive motion generation from examples,” *ACM Trans. Graph.*, vol. 21, no. 3, pp. 483–490, July 2002.
[11] J. Lee, J. Chai, P. S. A. Reitsma, and J. K. Hodgins, “Interactive control of avatars animated with human motion data,” *ACM Transaction on Graphics*, vol. 21, pp. 491–500, July 2002.
[12] A. Safonova and J. K. Hodgins, “Construction and optimal search of interpolated motion graphs,” in *ACM SIGGRAPH 2007 Papers*, 2007.
[13] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: Learning attractor models for motor behaviors,” *Neural Comput.*, vol. 25, no. 2, pp. 328–373, Feb. 2013.
[14] A. Safonova, J. K. Hodgins, and N. S. Pollard, “Synthesizing Physically Realistic Human Motion in Low-dimensional, Behavior-specific Spaces,” *ACM Transaction on Graphics*, vol. 23, pp. 514–521, 2004.
[15] B. Lim, S. Ra, and F. C. Park, “Movement primitives, principal component analysis, and the efficient generation of natural motions,” in *Proc. IEEE International Conference on Robotics and Automation*, 2005.
[16] J. Min, Y.-L. Chen, and J. Chai, “Interactive Generation of Human Animation with Deformable Motion Models,” *ACM Transaction on Graphics*, vol. 29, no. 9, 2009.
[17] J. O. Ramsay, G. Hooker, and S. Graves, *Functional Data Analysis with R and Matlab*. Springer, 2009.
[18] J. Aleotti, A. Cionini, L. Fontanili, and S. Caselli, “Arm gesture recognition and humanoid imitation using functional principal component analysis,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 3752–3758.
[19] W. Dai, Y. Sun, and X. Qian, “Functional analysis of grasping motion,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 3507–3513.
[20] R. Viviani, G. Gron, and M. Spitzer, “Functional principal component analysis of fmri data,” *Human Brain Mapping*, vol. 24, no. 2, 2005.
[21] A. Kassidas, J. F. MacGregor, and P. A. Taylor, “Synchronization of Batch Trajectories Using Dynamic Time Warping,” *AIChe*, vol. 44, no. 4, pp. 864–875, April 1998.
[22] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, “Experimental comparison of representation methods and distance measures for time series data,” *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, 2013.
[23] J. Serra and J. L. Arcos, “An empirical evaluation of similarity measures for time series classification,” *Knowledge-Based Systems*, vol. 67, pp. 305–314, 2014.
[24] G. Guerra-Filho and A. Biswas, “The human motion database, a cognitive and parametric sampling of human motion,” *Image and Vision Computing*, vol. 30, pp. 251–261, March 2012.
[25] MathWorks, “Optimization toolbox user’s guide r2014a,” 2014.
[26] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2006.
[27] I. Sucan, M. Moll, and L. Kavraki, “The open motion planning library,” *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec 2012.