Pattern-Preserving-based Motion Imitation for Robots

Bonggun Shin[1] and Sungho Jo[2]

[1]Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, South Korea
(Tel : +82-42-350-7740; E-mail: bgshin@kaist.ac.kr)
[2]Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, South Korea
(Tel : +82-42-350-3540; E-mail: shjo@kaist.ac.kr)

Abstract - This paper presents a new algorithm of encoding dynamic movements through pattern-preserving optimization by a physical robot. This research follows a recent robot programming approach called learning from demonstration in which the motion trajectory is learned from human demonstrations. The motivation of this work is to deal with major challenges in learning from demonstration such as embodiment mapping, generalization, adaptation, robustness to perturbations, stability, pattern-preserving, and parameter tuning. We propose a new method that can deal with those problems and present empirical results to support our insistence.

Keywords - learning from demonstration, motion imitation, pattern-preserving optimization

1. Introduction

Realization of flexible and adaptive behaviors for a robot in new situations is one of the most valuable features in designing of service robots. Because it is hard to preprogram for all possible tasks in robots and those preprogrammed robots can not adapt in real world situations where the environments change rapidly. In addition, in order for robots to be widely used, the programming process has to be easy and intuitive. Hence, recent studies have been focused on learning from demonstration [1, 2]. After a human teacher's demonstration is recorded, the similar trajectory is reproduced by a robot. Among those approaches, the dynamic movement primitive (DMP) framework[3, 4] considers imitations that can be generalized to new objectives, while other methods has limited ability in terms of generalization.

There are many known issues that should be dealt with to make this imitation learning practical. 1) Embodiment mapping problem[5] occurs when joints and links of a robot can not be directly matched with those of a demonstrator. 2) Generalization is desirable because users can not demonstrate all kinds of movements to make a robot perform various tasks. Learning from demonstration is not meaningful unless adapted reproductions are possible. For example, a demonstrated movement should be appropriately modified according to different start and goal positions. 3) Adaptation to unseen situations such as obstacle avoidance is required since it is not possible for a robot to predict all kind of situations when reproducing trajectories. In addition to adaptation, 4) robustness against perturbations is important because mere reproduction of the demonstrated trajectory would fail to reach a target position when external force changes the target position. 5) Stability should be taken into account for controlling robots for the following reason. If the robot does not stop at the goal position, the robot could break target a targeted object or harm humans nearby. 6) Parameter tuning process should not require engineering knowledges in order for the framework to be popularly used. If there is no parameter required to be tuned, robot programming will be easier. Over the past few years, a number of studies have been conducted on finding solutions of those issues. One of the recent approaches is the motion learning method based on Gaussian mixture model (GMM) to ensure generalization, stability, parameter adaptation and robustness to perturbations [6, 7]. Their methods are able to generalize for unseen contexts. However, the generalization can be assured only in the limited region which is defined around the demonstrated trajectory. The limited generalization is a critical problem because learning from imitation is feasible only if replays are generated not in specific contexts but in general contexts. Another recent research is based on DMP[8]. They proposed the modified and extended version of DMP to ameliorate conventional DMP's undesired behaviors such as overshoots. Although it ensures global stability and generalization, it is hard to find the appropriate parameter and stability and pattern-preserving property can not be satisfied at the same time.

To master these problems, we present a new framework of motion imitation that is based on the pattern-preserving optimization. The motion optimization approach was proposed in the literature[9], and the optimization variables are in joint space. Instead of joint space trajectories we concern task space trajectories which directly capture the patterns of a motion. In addition, we use the second derivatives of a trajectory curve as a optimization variable, while most of past methods optimize the position of a trajectory. This is because the second derivatives can be seen as curvatures in one dimension, which contain the shape of a trajectory and are spatially invariant. In this paper, we consider a simple component of a motion, called the point-to-point motion, that can be combined to the complex motions. This simple motion modeling is beneficial since learning from demonstration concept can be easily applied. The proposed method reproduces simple point-to-point motions that preserve the demonstrated motions patterns, while settling the problems mentioned above.

The remaining sections are structured as follows. In Section II, we propose our method, motion trajectory morphing(MTM) for one dimensional trajectory, and

978-1-4577-0723-0 / 11 / $26.00

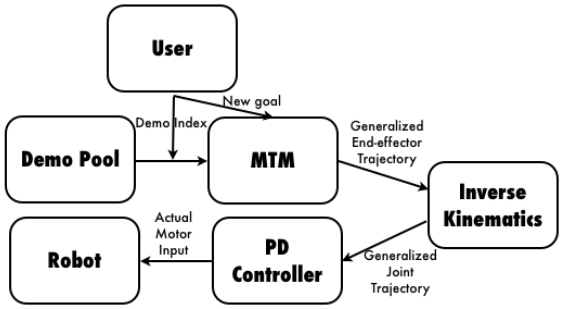


Fig. 1 System Overview

added the obstacle avoidance module. Section III discuss robustness against spatial and temporal perturbations. The stability and pattern-preserving problem are considered in Section IV. We show feasibility of our method empirically in Section V, before conclude this approach in Section VI.

2. Trajectory Morphing for General Reproductions

We represent a motion trajectory in operational space[10] to deal with embodiment mapping problem. Usually a robot's movement can be represented with six dimensions in task space, three dimensions for a position of the end effector in the Cartesian coordinate and the other three for its orientation information. In our robot experiments, we use inverse kinematics to calculate the appropriate joint angles from the trajectories of end-effector position and orientation. Since we develop our method in operational space, we are free from the embodiment mapping problem.

2.1 System Overview

A single MTM is applicable to a one-dimensional system. When the six dimensional operational space is concerned, each dimensional trajectory is morphed by each MTM. The reproduction procedure of a generalized trajectory is as follows. The human teacher demonstrate a point-to-point motion kinesthetically, in other words the teacher holds robot's arm and move it to perform a specific motion. Recording the six dimensional trajectories takes place simultaneously. This can be done either using a motion capture device or reading encoder values. When a new start and goal positions are specified, the Algorithm 1 calculates the new trajectory that starts from the new starting points and ends at the new goal points with having the similar shapes in the middle of the trajectory. Then we convert the trajectory in task space into the joint space trajectory using inverse kinematics. The actual motor control inputs are calculated using PD controllers. The whole procedure is shown in the Fig. 1.

2.2 Motion Trajectory Morphing

In this paper, we assume independence of each dimension of a trajectory, so we consider each dimension's curve separately.[1] We consider the teacher's demonstration trajectory which is represented as a curve(parameterized by time t), d_t, $t = 0 \cdots T-1$ (equivalently $\mathbf{d}$ in vector notation). The problem is to find a pattern preserved curve, x_t, $t = 0 \cdots T-1$ (equivalently $\mathbf{x}$ in vector notation), for given starting and goal point constraints, $x_0^{'}$ and $x_{T-1}^{'}$, respectively. We define dissimilarity of two curves as a cost function and optimize it to find a similar curve morphed from the straight line that crosses from $x_0^{'}$ to $x_{T-1}^{'}$.

A. Similarity Metric

The essential feature of imitation learning is to preserve the shape of the demonstrated trajectory. Therefore an imitating trajectory should have the similar shape with the demonstrated trajectory. To produce an similar generalized trajectory, we need to measure the similarity of two trajectories. The similarity of given two curves can be measured by comparing their second derivatives for the following reason. The shape of a curve can be captured by the set of curvatures at each point. For a given curve, the curvature at a specific point is defined by the rate of change of the unit tangent vector. By assuming each trajectory dimension is independent, the curvature becomes effectively the rate of change of its velocity, which is its second derivative.

B. Dissimilarity of curves

To formulate a form of the optimization problem, we need to define a cost function. Since the objective is to find the similar curve, the cost function is defined using the dissimilarity of curves, the difference of the second derivatives. We define the dissimilarity, $D_t(\mathbf{d},\mathbf{x})$, for given any two curve, $\mathbf{d}$ and $\mathbf{x}$ by (1).

$$D_t(\mathbf{d},\mathbf{x}) = \kappa_{d_t} - \kappa_{x_t}, \tag{1}$$

where κ_{d_t} and κ_{x_t} are the second derivatives of the demo and the objective trajectories at time t, respectively.

The objective is to minimize the cost function as following

$$\min_{\mathbf{x}\in\mathbb{S}} J(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{S}} \Sigma_{t=1}^{T-2} D_t(\mathbf{d},\mathbf{x})^{\mathbf{2}} \tag{2}$$

The first and the last elements of each vector are not used because the second derivatives can not be defined at the first and the last points (Note that the summation in (2) excludes those points).

The trajectories are defined in the specific task space, and represented as $\mathbb{S}$. It represents not only robot's physical boundaries but environmental boundaries such as obstacles. These boundaries can be recognized in advance to be used as the lower bounds, $\mathbf{l}$ and the upper bounds, $\mathbf{u}$.

C. Algorithm

For a given demo trajectory, $\mathbf{d}$, the optimization of the cost function (2) finds the similar trajectory, $\mathbf{x}$, starting

[1] The DMP also assumes the independence of each trajectory dimension, because it is easier to integrate all dimensional trajectories.

Algorithm 1 Motion Trajectory Morphing (MTM)

Input: $\mathbf{d}, x_0^{'}, x_{T-1}^{'}, \mathbf{l}, \mathbf{u}$

Initialize $x_{0:T-1}$ to be the straight line crossing from $x_0^{'}$ to $x_{T-1}^{'}$
Initialize ε with an appropriate small number
set l_t and u_t according to $\mathbb{S}$
$J_{old}(x) \leftarrow 0$
$T \leftarrow$ trajectory length
repeat
  $J(\mathbf{x}) \leftarrow \sum_{\mathbf{t=1}}^{\mathbf{T-2}} \mathbf{D_t}(\mathbf{d},\mathbf{x})^2$, using (1)
  $\mathbf{x} \leftarrow$ L-BFGS-B$(J(\mathbf{x}), \mathbf{J_{old}}(\mathbf{x}), \mathbf{x_0^{'}}, \mathbf{x_{T-1}^{'}}, \mathbf{l}, \mathbf{u})$
  $\delta \leftarrow |J(\mathbf{x}) - \mathbf{J_{old}}(\mathbf{x})|$
  $J_{old}(\mathbf{x}) \leftarrow \mathbf{J}(\mathbf{x})$
until $\delta \geq \varepsilon$
Output: $\mathbf{x}$

from the new start point and heading toward the new goal point. To fix the boundary points, the optimization runs with those points excluded. The algorithm for one dimensional MTM is shown in Algorithm 1.

Since the optimization problem is nonlinear and has the boundary condition, it can be solved by the famous nonlinear optimization solver, the L-BFGS-B algorithm[11].

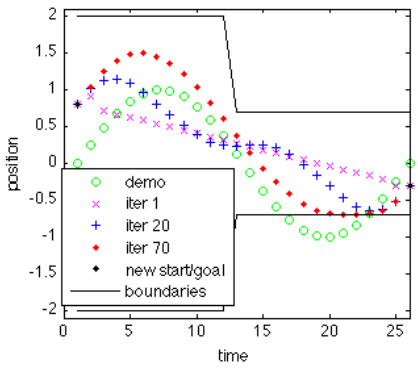


Fig. 2 Iterations of the MTM algorithm. The demo trajectory(blue dots) is the sine curve from 0 to 0. The optimization is done with the new starting point, 0.8 and goal point, -0.3. The trajectory has bound (-2,2) for the first half, and (-0.7,0.7) for the second half. Note that the optimization result was straight line at the beginning of the iteration (magenta x-marks), but it is morphed continuously to the similar looking trajectory(red stars) as the iteration goes by, with satisfying boundary constraints(black crosses).

Figure 2 shows the optimization process for each iteration. The demo trajectory is the sine wave. The new trajectory is generated by MTM for the given new starting point, 0.8 and the new goal point, -0.3. We can see that the new trajectory meets the boundary constraints and contains the similar shapes in the curve. This shows that MTM is generalizable and preserves patterns as well. However, GMM based approaches has generalization capability only in the small region that is not far from the demonstrated area[7]. DMP based approaches can not satisfy stable generalization and pattern preserving requirement at the same time.[2]

[2]Section III discusses about stability of MTM, GMM, and DMP in detail.

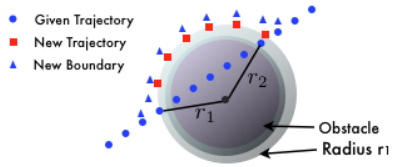


Fig. 3 New trajectory calculation

2.3 Obstacle Avoidance

We extend MTM to be capable of obstacle avoidance. To consider the avoidance behavior, the boundary conditions, **L** and **U** are modified as described bellow. (We use matrix notations of **L** and **U** to represent 3D dimensional boundaries)

If we assume an obstacle as a sphere, we can represent the obstacle as a tuple $(c^{(1)}, c^{(2)}, c^{(3)}, r)$. $c^{(1)}$, $c^{(2)}$ and $c^{(3)}$ represent the 3D location of the center of the obstacle and r is the radius that encloses the obstacle.

If the computed trajectory goes through the obstacle (blue circles in the Fig. 3), we have two points that are not collided and adjacent to the collided points. The farther point, measured from the center of the obstacle, is selected as safe boundary (red squares in the Fig. 3). This boundary is updated using sphere rotation function (17th line in the Algorithm 2). Then we call MTM(Algorithm 1) to recompute sub-trajectories(blue triangles in the Fig. 3) with the new boundaries.

Algorithm 2 Obstacle Avoidance (Avoid)

Input: $\mathbf{X}, \mathbf{D}, \mathbf{L}, \mathbf{U}$ $(c^{(1)}, c^{(2)}, c^{(3)}, r)$

if $\mathbf{X}$ crosses the obstacle $(c^{(1)}, c^{(2)}, c^{(3)}, r)$ **then**
  $t_{o_1} \leftarrow$ the time index of the initial cross -1
  $t_{o_2} \leftarrow$ the time index of the end of the cross $+1$
end if
  $\mathbf{v_1} \leftarrow (\mathbf{X}^{(1)}_{\mathbf{t_{o_1}}} - \mathbf{c^{(1)}}, \mathbf{X}^{(2)}_{\mathbf{t_{o_1}}} - \mathbf{c^{(2)}}, \mathbf{X}^{(3)}_{\mathbf{t_{o_1}}} - \mathbf{c^{(3)}})$
  $\mathbf{v_2} \leftarrow (\mathbf{X}^{(1)}_{\mathbf{t_{o_2}}} - \mathbf{c^{(1)}}, \mathbf{X}^{(2)}_{\mathbf{t_{o_2}}} - \mathbf{c^{(2)}}, \mathbf{X}^{(3)}_{\mathbf{t_{o_2}}} - \mathbf{c^{(3)}})$
  $\mathbf{n} \leftarrow \mathbf{v_1} \times \mathbf{v_2}$, $r_1 \leftarrow |\mathbf{v_1}|$, $r_2 \leftarrow |\mathbf{v_2}|$, $r_{sel} \leftarrow \max(r_1, r_2)$
  if $r_1 \geq r_2$ **then**
    $\mathbf{v_{sel}} \leftarrow \mathbf{v_1}$
  else
    $\mathbf{v_{sel}} \leftarrow \mathbf{v_2}$
  end if
  $N \leftarrow t_{o_2} - t_{o_1}$
  $\theta \leftarrow \arccos(\frac{\mathbf{v_1} \cdot \mathbf{v_2}}{|\mathbf{v_1}||\mathbf{v_2}|})$
  $\Delta\theta \leftarrow \theta / N$
  for all $0 \leq i \leq N-1$ **do**
    $\mathbf{v_{rot}} \leftarrow \mathbf{Rotate^n}(\mathbf{v_{sel}}, \triangle\theta_\mathbf{i})$
    for all $1 \leq j \leq 3$ **do**
      if $X^{(j)}_{t_{o_1}+i} \geq c(j)$ **then**
        $\mathbf{L^{(j)}_{t_{o_1}+i}} \leftarrow \mathbf{v^{(j)}_{rot}}$
      else
        $\mathbf{U^{(j)}_{t_{o_1}+i}} \leftarrow \mathbf{v^{(j)}_{rot}}$
      end if
    end for
  end for
  Set w to include $2w$ more points for optimization
  $t_{o_1} \leftarrow t_{o_1} - w$, $t_{o_2} \leftarrow t_{o_2} + w$
  for all $1 \leq j \leq 3$ **do**
    $\mathbf{X^{(j)}_{t_{o_1}:t_{o_2}}} \leftarrow$ MTM $(\mathbf{D^{(j)}_{t_{o_1}:t_{o_2}}}, \mathbf{X^{(j)}_{t_{o_1}}}, \mathbf{X^{(j)}_{t_{o_2}}}, \mathbf{L^{(j)}_{t_{o_1}:t_{o_2}}}, \mathbf{U^{(j)}_{t_{o_1}:t_{o_2}}})$
  end for
Output: $\mathbf{X}$

Algorithm 3 Remedy for Perturbations (SP)

Input: $s, t_s, \mathbf{d}$

$T^{'} \leftarrow T - t_s$
$x_0^{'} \leftarrow x_{t_s}, x_{T^{'}-1}^{'} \leftarrow x_{T-1}^{'} + s$
for all $0 \leq t \leq T^{'} - 1$ **do**
  $d_t^{'} \leftarrow d_{t_s+t}$
  $l_t^{'} \leftarrow l_{t_s+t}, u_t^{'} \leftarrow u_{t_s+t}$
end for
$\mathbf{x}^{'} \leftarrow$ MTM $(\mathbf{d}^{'}, x_0^{'}, x_{T^{'}-1}^{'}, \mathbf{l}^{'}, \mathbf{u}^{'})$

Output: $\mathbf{x}^{'}$

3. Robustness to Perturbations

Perturbations can be classified into spatial perturbations and temporal perturbations[6]. Spatial perturbations affect the position of the robot in task space and temporal perturbations are related to the planned motion duration. Both GMM and DMP based methods are robust to perturbations, In this section, we present the algorithm for robustness to perturbations.

3.1 Spatial Perturbations

Spatial perturbations occur when goal position is changed after the onset of a motion and yet the trajectory changes do not require a delay in time. The MTM can be adapted to spatial perturbations with a minor correction as follows. Once the spatial perturbations occur after the onset of a motion, the new MTM is executed with modified inputs. For example, the perturbation occurred at the goal in $t = t_s$ with the displacement, s. For the new MTM, we use the new demonstration as $d_{t_s}, d_{t_s+1}, \cdots, d_T$, and the new goal position as $x_{T-1}^{'} + s$. This procedure is summarized in Algorithm 3.

3.2 Temporal Perturbations

Temporal perturbations occur when the goal position is changed after the onset of a motion and this results in a delay to approach the target point. This is almost the same with spatial perturbations except that the temporal perturbations require more execution time to finish the trajectory. The new goal constraint can be met by using Algorithm 3, but if the generated trajectory requires more time to reach the target(temporally Perturbed), it might produce jerky motions. To cope with this problem, the delay(Δt in Algorithm 4) is calculated so that it is used in the adaptive PD controller. Algorithm 4 integrates all modifications mentioned above and designed for actual robot systems(Fig. 1). (Note that trajectories are represented as matrices due to multidimensionality.)

4. Stability versus Characteristic Preserving

Here, we consider stability/pattern-preserving dilemma. One of the important purposes of the learning from demonstration is imitation of the demonstrated patterns. Because if it fails to preserve the essential patterns of the demonstrated motion, then the reproduced motion is distorted or misrepresented so it can not be regarded as the imitation. In addition, motion reproducing systems have to be stable. To represent a complex motion, the system synthesizes it from a set of simple point-to point motions. Each point-to-point motion should approach and stop at the exact goal position, in order to perform a task smoothly when combined together. Therefore stability is also inevitable property of the motion reproducing systems. Previous methods can not satisfy both of these two requirements at the same time, and we name this problem as the stability/pattern-preserving dilemma. In this section, we describe why our method are free from the stability/pattern-preserving dilemma.

Algorithm 4 MTM for Robot Movements

Input: $\mathbf{D}, \mathbf{X_0}, \mathbf{X_T}, \mathbf{L}, \mathbf{U}, t_{desired}$

for all $1 \leq j \leq N$ **do**
  $\mathbf{X^{(j)}} \leftarrow$ MTM $(\mathbf{D^{(j)}}, \mathbf{X_0^{(j)}}, \mathbf{X_T^{(j)}}, \mathbf{L^{(j)}}, \mathbf{U^{(j)}})$
end for
if An obstacle exists with position (a,b,c) and radius r **then**
  $\mathbf{X} \leftarrow$ Avoid($\mathbf{X}$, $\mathbf{D}$, $\mathbf{L}$, $\mathbf{U}$, (a,b,c,r))
end if
$\mathbf{E} \leftarrow$ InversKinematics($\mathbf{X}$)
$\Delta t \leftarrow t_{desired}/T$
for all $0 \leq t \leq T$ **do**
if Perturbations **then**
  $dist_{old} \leftarrow \max_{1 \leq j \leq N}(E_t^{(j)} - E_t^{(j)})$
  $\mathbf{s} \leftarrow$ Perturbation displacement
  for all $1 \leq j \leq N$ **do**
    $\mathbf{X_{t:T}^{(j)}} \leftarrow$ SP($\mathbf{s^{(j)}}, t, \mathbf{D^{(j)}}$)
  end for
  $\mathbf{E_{t:T}} \leftarrow$ InversKinematics($\mathbf{X_{t:T}^{(j)}}$)
  $dist_{new} \leftarrow \max_{1 \leq j \leq N}(\mathbf{E_t^{(j)}} - \mathbf{E_T^{(j)}})$
  $\Delta t \leftarrow \Delta t \cdot dist_{new}/dist_{old}$
end if
PDController($\mathbf{E_s(t)}, \Delta t$)
end for

Output: X_{new}

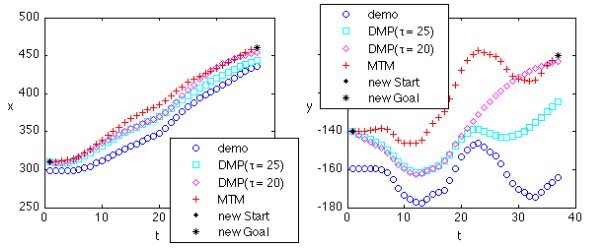
4.1 Stability at Target

Stability condition at the goal position is considered as one of the desiderata of the motion generation system[7]. To deal with stability issue in GMM based methods, BM algorithm[7] is proposed. They provided the conditions that guarantee asymptotic stability of the generated trajectory, but DMP provides the better solution. If we use big τ, then the effect of pattern-preserving term is suppressed that ensures to reach the goal position(stability). However, for MTM, since the optimization process is executed with fixed start and goal position, the reproduced trajectory is always stable, as long as the demonstrator provides a stable demonstration.

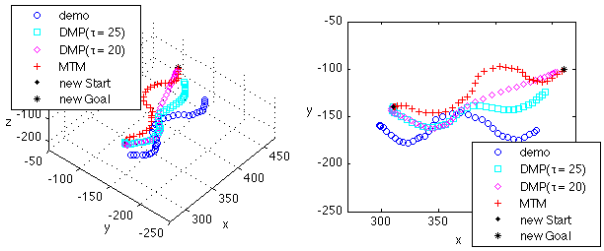
4.2 Characteristic Preserving

BM[7] satisfies stability requirement, but it provides stability at the sacrifice of pattern preserving. Outside of the specific region, its pattern is ignored to satisfy the stability. DMP is devised to consider both of these features by coupling two terms, a stable linear dynamic system and an estimator of the demonstrated trajectory. The effects of these two terms are controllable by changing a

parameter, τ. Big τ produces a pattern preserved trajectory but falls short of satisfying stability. Small τ guarantees the stability of the trajectory while demonstration patterns are disappeared around the goal position. So it is hard to find a good τ and sometimes there is no τ that satisfies both of those features at the same time(Fig. 4). However, when we use MTM, pattern information is preserved well because the cost function of the optimization is based on the second derivatives of trajectories. Therefore the MTM based motion reproducing systems satisfy stability and pattern preserving without having a parameter tuning task. The comparison result is shown in the Fig. 4.



(a)1D Trajectory, $x(t)$ (b) 1D Trajectory, $y(t)$

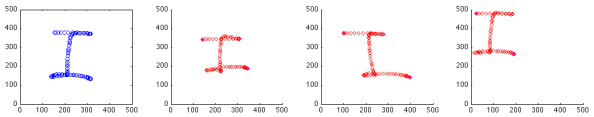


(c) 3D Trajectory, $(x(t), y(t), z(t))$ (d) 2D Trajectory, $(x(t), y(t))$

Fig. 4 The demo trajectory(blue circles) was kinesthetically given by human using Katana arm. For the given new constraints(black dots and stars), the comparison results are shown. The trajectory generated by MTM(red crosses) satisfies both the pattern-preserving property and the stability at the target, while DMP curves put more weights on only one of them according to the τ values.

5. Experiment Results

For the first part of the experiments, we investigate the proposed framework through 2D data created by a Tablet PC, and this experiment is firstly proposed by Khansari-Zede's[7] to examine the pattern preserving property. The demonstration data was created by a normal person with a Tablet PC, and he chose the new start and goal positions, obstacle position, and perturbed target positions. This experiment shows generalization, stability, obstacle avoidance, and robustness to perturbations of the proposed framework through the computer simulation. We further describe how we applied the proposed framework on the physical robot, the Katana arm to validate our method. Katana arm has six DOF, and the demonstration was given to the system kinesthetically. Obstacle positions and perturbed target positions are tracked by a



(a)Demo 'I' (b)Output1 'I' (c)Output2 'I' (d)Output3 'I'

Fig. 5 Generalization Test on Tablet PC. 4 examples are shown. Blue graphs represent demonstration and the red graphs are reproduction results with various start/goal positions.

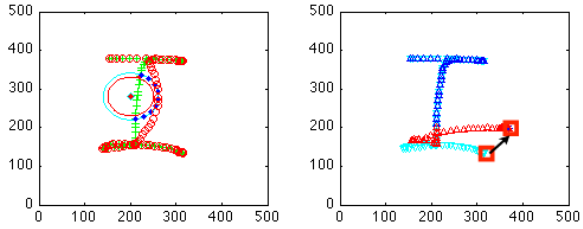
single camera for simplicity.[3] From this physical robot experiment, we also ensure that our framework has obstacle avoidance and robustness to perturbations in real robot situations.

The first part of the tablet PC experiment is aimed to see pattern preserving generalization to new start and target points (Fig. 5) The demonstration trajectories (Fig. 5(a)) are recorded using the tablet PC by a human, and that person is asked to select start and end points. Using these information, our algorithm found generalized trajectories successfully (the three red figures in the Fig. 5). The second part of of the tablet PC experiment is to validate the Algorithm 2. We followed the similar procedure with the previous part of the experiment except that the person is asked to select the position of the obstacle instead of start and end position. The algorithm 2 shows the capability of obstacle avoidance (Fig. 6(a)). Since the tablet screen is two dimensional, an obstacle is modeled as a circle instead of a sphere and the Algorithm 2 was modified appropriatly. For three dimensional case, we can just use the Algorithm 2.

The adaptation algorithm to perturbation is also tested using the tablet PC (Fig. 6(b)). The perturbation occurring time is set to $T/2$, and this assumption is released in the physical robot test.[4] At time $T/2$, the goal point is changed, and the Algorithm 3 recalculate the rest of the trajectories (marked as the red triangles in the Fig. 6(b)). Even if the end point is changed the shape of the trajectories are well preserved.

The simulation results tell us that the proposed algorithms produce the expected results. We further carried out physical robot experiments to validate our method in the real world situations. The tasks was the drawing a letter task which is the same with the tablet PC experiment. But the actual robot experiments consider not only 3D space movements, but also orientations of the end-effector. That means that we can reproduce learned skills in other areas.[5] Similar to the tablet PC experiment, we carried out three kinds of experiment with the Katana arm. The demonstration was given to the robot by a person's hand (Fig. 7(a)). If we let the robot know the start and goal points, then it was able to reproduce the demo adapted to that situation (Fig. 7(b)). The repro-

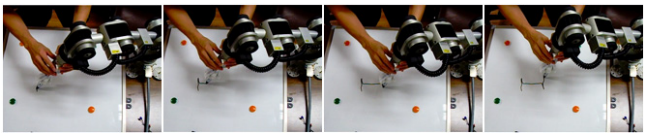
[3]Note that we used a 2D obstacle to simplify the vision system, but the robot was ran in the 3D space actually.
[4]The physical robot system uses vision that tracks when perturbation occurs.
[5]The drawing a letter task should consider robot hand's orientation, because the robot write with a marker on the white board.



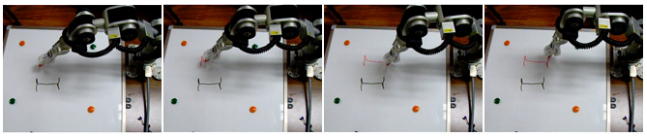
(a)'Obstacle avoidance' (b)'Adaptation to perturbations'

Fig. 6 6(a): The green crosses represent original trajectories and the small red circles are the results of obstacle avoidance. Obstacles are represented by the big inner circle(red) and the big outer circle(cyan) shows the new boundary for optimizations. 6(b): The perturbation is caused by the change of the goal position. To clarify the time when the perturbation occurs the graph uses different colors for the trajectories before the perturbation (blue triangles) and after the perturbation. One can see that the adapted trajectories(red triangles) are successful in that they preserve demonstration(cyan bottom-up triangle) pattern and keep following the goal positions.

ductions were successful even if the obstacle was existed (Fig. 8(a)) and a perturbation is applied to the target (Fig. 8(b)).

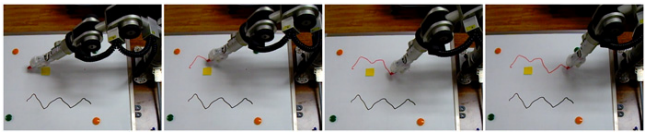


(a)Teaching the robot (demonstration)

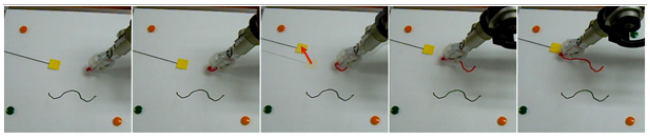


(b)Imitation with the new start and goal points

Fig. 7 Generalization Test on Robot



(a)Reproduction with obstacle avoidance. The yellow square was placed in the middle of the new trajectory. The robot successfully avoided the obstacle and preserved shapes of the demonstrated trajectory



(b)Reproduction with adaptation to the perturbation. The target point is the yellow square in this experiment. The target point was changed after the robot had started the reproduction of the demo motion. The robot successfully recalculated the adapted trajectory and reached the new target.

Fig. 8 Robustness to Environment Changes

6. Conclusions and Future Works

In this paper, we proposed the new framework for reproducing generalized motion using the pattern preserving optimization. We presented several algorithms that each of them is a submodule of the proposed framework. The proposed method was shown to be able to deal with major challenges in learning from demonstration. It can generalize to new situations without parameter tuning, and it has adaptation capability to changed environments such as obstacles and perturbations of a target position. Our framework was tested through the simulation and then applied to the physical robot.

We demonstrated generalization of one point-to-point motion, writing a letter 'I'. Once the robot learns how to write all the alphabets as a set of separate point-to-point motions, then the robot can write any combinations of alphabets in any place.

The proposed framework can be further extended when we use multi-dimensional curvatures. Since we assume each trajectory dimension is independent, sometimes the proposed method shows distorted shapes in terms of orientation If we use multi-dimensional curvatures, those effect will be disappeared.

Acknowledgement

This research was supported in part by the MEST, Korea, under the KRF grant (No. 2010-0015226) and in part by the MKE, Korea, under the Human Resources Development Program for Convergence Robot Specialists support program.

References

[1] S. Schaal, "Is imitation learning the route to humanoid robots," *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] A. Billard and R. Siegwart, "Special issue on robot programming by demonstration," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, 2004.

[3] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings- IEEE International Conference on Robotics and Automation*, vol. 2. Citeseer, 2002, pp. 1398–1403.

[4] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 358, no. 1431, p. 537, 2003.

[5] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[6] E. Gribovskaya, K. Zadeh, S. Mohammad, and A. Billard, "Learning Nonlinear Multivariate Dynamics of Motion in Robotic Manipulators," *International Journal of Robotics Research*, 2010.

[7] S. Khansari-Zadeh and A. Billard, "BM: An Iterative Algorithm to Learn Stable Non-Linear Dynamical Systems with Gaussian Mixture Models," *In Proceeding of the International Conference on Robotics and Automation (ICRA)*, pp. 2381–2388, 2010.

[8] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," *ICRA, Japan, May*, pp. 12–17, 2009.

[9] B. Lim, S. Ra, and F. Park, "Movement primitives, principal component analysis, and the efficient generation of natural motions," apr. 2005, pp. 4630 – 4635.

[10] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, p. 737, 2008.

[11] C. Zhu, R. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.