Learning to Play Minigolf: A Dynamical System-based Approach

S.M. Khansari-Zadeh Klas Kronander Aude Billard

Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland

{*mohammad.khansari,klas.kronander,Aude.Billard*}*@epfl.ch*

Abstract

A current trend in robotics is to define robot motions so that they can be easily adopted to situations beyond those for which the motion was originally designed. In this work, we show how the challenging task of playing minigolf can be efficiently tackled by first learning a basic hitting motion model, and then learning to adapt it to different situations. We model the basic hitting motion with an autonomous Dynamical Systems (DS), and solve the problem of learning the parameters of the model from a set of demonstrations through a constrained optimization. To hit the ball with the appropriate hitting angle and speed, a nonlinear model of the hitting parameters is estimated based on a set of examples of good hitting parameters. We compare two statistical methods, Gaussian Process Regression (GPR) and Gaussian Mixture Regression (GMR) in the context of inferring the hitting parameters for the minigolf task. We demonstrate the generalization ability of the model in various situations. We validate our approach on the 7 Degrees of Freedom (DoF) Barrett WAM arm and 6-DoF Katana arm in both simulated and real environments.

keywords: Hitting motions, Imitation Learning, Nonlinear dynamical systems, Playing minigolf

1 Introduction

The traditional approaches to controlling robots, by explicitly defining tasks by hand-coding them, is ill-suited for bringing robots into unstructured environments. Consequently, different approaches to transferring skills to robots have been proposed. One such approach is imitation learning (also known as programming by demonstration) [1], where tasks are demonstrated to a robot by an expert.

An important concept in imitation learning is the ability to generalize the task and to adapt it to a new situation. This concerns the problem of performing the task under different circumstances than those present during demonstrations, which is desirable mainly for two reasons: 1) The number of demonstrations can be kept small, and 2) Given appropriate adaptation, an acquired skill can be used to carry out a more complex task than the teacher is capable of demonstrating. Dynamical Systems (DS) provide a powerful tool for robust control of robot motions from a small set of demonstrations [2]. They ensure high precision in reaching a desired target, yet can be easily modulated to generate new motions in areas not seen before [2–4].

The minigolf[1] task is a typical case covered by the motivations of the adaptation presented above. In [5], it is shown that human players in ball games such as minigolf usually follow the same pattern of motion when approaching the ball, even if circumstances such as ball position change. The goal in minigolf is to sink[2] the ball into a hole located on the field. There are often obstacles and curved surfaces between the ball and the hole, to make the task more complex for the player. To play this game, a player needs first and foremost to learn how to swing the golf club so as to hit the ball precisely. Additionally, depending on the situation, the ball may have to be hit at a particular angle and speed. Human players can achieve this by rotating their body prior to hitting the ball, and adapting the hitting speed. In this work, we follow a similar two-steps approach for teaching minigolf to a robot in which: a) We first learn the basic motion for hitting the ball, and 2) we then learn to use the correct hitting angle and speed for the given location of the ball.

For the first subtask, we consider a hitting motion that is modeled with autonomous (i.e. time-invariant) DS. We model an estimation of this DS with a Gaussian Mixture Model (GMM), and propose a constrained optimization problem to learn the parameters of the model from a set of demonstrations. Specifically, we extend the previous formulation of our nonlinear autonomous DS [2] to model robot motions with *a desired velocity* at the target. This extension allows to learn a considerably wider set of motions ranging from pick-and-place movements to agile robot tasks that require reaching/hitting a given target with a specific speed and direction. For the second subtask, the hitting parameters (angle and speed with which to hit the ball) are learned from a training set collected with the aid of a teacher specifying good values for some different hitting locations. We use two statistical methods (GMR [6] and GPR [7]) to infer hitting parameters for unseen hitting locations. The performance of the proposed approach is evaluated in robot experiments of playing minigolf on different challenging fields using the 7-DoF Barrett WAM arm and 6-DoF Katana arm, both equipped with a golf club tool.

2 Related Work

In imitation learning, robots are taught to perform a task by observing a set of demonstrations provided by a teacher (human or robot). Demonstrations to a robot may be performed in different ways; back-driving the robot, teleoperating it using motion sensors, or capturing a task via vision sensors. The learning process consists of extracting the relevant information from the demonstrations and encoding this information in a motion model that can be used to reproduce the task. Using a set of basic motion models (also commonly referred to as motion primitives) learned in this way, more advanced robot motions can be achieved by combining and adapting the different motion models.

Different motion models and subsequently different learning techniques have been proposed, including but not limited to: polynomial and spline-based methods [8–10], nonlinear regression techniques [11, 12], time-dependent DS [13, 14], and autonomous DS [2, 3, 15]. These methods have been successfully developed to learn motion primitives such as discrete (point-to-point) motions [2–4, 10–15] and their

[1] Also commonly referred to as mini-golf, miniature golf, midget golf, crazy golf and Putt-Putt.

[2] Sinking means hitting the ball such that it goes into the hole.

extensions to obstacle avoidance [14,16], rhythmic motions [4,13], hitting motions [15,17,18], etc. This paper focuses on two problems: 1) the learning of hitting motions (i.e. robot trajectories with non-zero velocity at the target), and 2) their adaptation to different situations. Next, we review approaches that tackle either of these problems with an emphasis on those that exploit machine learning algorithms.

One of the preliminary works on learning a DS model of human-like motions through imitation learning is presented in [4]. There, the authors used a Dynamic Movement Primitive (DMP) model to lightly touch a ball in a tennis swing motion. In [17], a modification to the original formulation of DMP is proposed to generate striking motions in table-tennis. In this approach, the desired hitting speed and direction are obtained by considering a moving virtual target. In our previous work [15], we considered an alternative DS approach based on Hidden Markov Model (HMM) and GMR. The method presented there is used to generate two different strokes in table tennis. Besides the approach using imitation learning, there are other approaches that focus on generation of feasible trajectories based on an estimation of ball's position [8,9,19]. In these works, given a robot arm's initial position and a desired final point, the hitting motion is modeled either as a fifth order polynomial of time [8,9] or the shortest path that is determined by optimizing the task-based directional manipulability measure [19].

The adaption of the learned motion models to different situations has previously been studied taking a Reinforcement Learning (RL) approach in [20]. The authors propose the Cost Regularized Kernel Regression (CRKR) algorithm which learns task-appropriate parameters for motion primitives. Impressive results from learning dart and table-tennis hitting with the 7-DoF Barrett WAM are presented. The robot autonomously explores the parameter space and learns how to adapt to new situations through trial and error. In [21], they use reinforcement learning with function approximation to learn when to start tilting the bottle in a pouring task, depending on the glass location. Another interesting work is [22], which presents an integrated approach to teach the skill of archery to a humanoid. Assuming that the basic elements of the task are known (i.e. shooting an arrow) the robot autonomously adapts this basic policy so that the center of the target is hit. In [23] an algorithm to optimize the whole robot trajectory for each new situation based on a set of demonstrations is proposed.

Our work differs from all of these in that we take an imitation learning approach to model both the basic hitting motion and its adaptation to different situations. While this frees us from having to define a task-specific cost function, it requires the availability of a teacher that can provide the training data. Furthermore, we consider a time-invariant approach to model the hitting motion which provides us with an inherent robustness to perturbations. In addition, this formulation does not require planning in advance or re-planning in the face of perturbations, and is suited for real-time implementation. Regarding the adaptability to different situations, we contrast the generalization abilities of GMR and GPR, two techniques widely used in robotics. The work presented here was published in a preliminary form in [18]. The present paper expands on our previous work in three ways: a) it provides a more detailed description of the hitting motion, b) it proposes an optimization problem to efficiently build an estimate of the hitting motion from demonstrations, and c) it presents more robot experiments, verifying the generalization ability of the presented approach and its robustness to perturbations.

3 Problem Statement

In the minigolf task considered in this work, the player only gets one chance to sink the ball. To achieve this, first of all, the player must know how to swing the golf club to hit the ball. This requires having general knowledge about how to hit the ball in various situations depending on the position of the player, the ball, and the hole. For example consider Fig. 1a. In this example, there are several initial positions where the robot is required to start a swing motion and hit the ball along a desired direction. This is a non-trivial task, which cannot be simply fulfilled by just playing recorded trajectories. Additionally, playing in dynamic environments where the ball or the hole could be displaced during the swing phase requires an online and smooth adaptation of the swing motion in order to fulfill the task and to sink the ball. In this paper we consider a DS approach to model the hitting motion. When encoding the hitting motion with an autonomous DS, this problem reduces to estimating a smooth first order differential equation $\boldsymbol{f}_h(\boldsymbol{x})$:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_h(\boldsymbol{x}) \qquad \boldsymbol{f}_h : \mathbb{R}^3 \mapsto \mathbb{R}^3 \tag{1}$$

When controlled through a Dynamical System (DS), a robot motion unfolds in time. Given an initial point $\boldsymbol{x}^0$, the robot motion along time can be computed by integrating $\boldsymbol{f}_h(\boldsymbol{x})$ through time. The main challenge in estimating $\boldsymbol{f}_h(\boldsymbol{x})$ is to ensure that starting from any initial point $\boldsymbol{x}^0 \in \mathbb{R}^3$, the temporal evolution of the motion passes through the target point (i.e. hits the ball) at a desired speed and direction, while retaining the main features presented in demonstrations. We will address this problem in Section 4.

Now assume the player has learned a planar hitting motion and can hit the ball in a direction specified by the unit vector $\boldsymbol{\psi}^* \in \mathbb{R}^2$ in the hitting plane and with hitting speed $v^* \in \mathbb{R}^+$ (see Fig. 1b). For each new situation, a hitting angle α and hitting speed gain κ must be chosen such that hitting with speed κv^* in direction $\boldsymbol{\psi}^*_\alpha = \boldsymbol{R}_\alpha \boldsymbol{\psi}^*$ (where R denotes a counterclockwise rotation by α in the hitting plane) leads to sinking the ball. Estimating these parameters is a potentially very hard task for advanced fields.

Consider the simplest possible minigolf field: a flat field without obstacles. Such a field is depicted in Fig. 1b. In this case the choice of hitting angle is trivial - the ball should simply be hit in a straight line towards the hole. The vector $\boldsymbol{s} \in \mathbb{R}^2$ denotes the relative position of the hole to the ball projected in the hitting plane. This vector represents the *situation* that the player has to adapt to when choosing the hitting parameters. As can be seen in Fig. 1b, to play the flat field, the player simply has to align the hitting direction $\boldsymbol{\psi}^*_\alpha$ with this vector. With the correct hitting angle, the player can use a wide range of speeds that result in sinking the ball.

Now consider the more advanced field such as the arctan field[3] (see Fig. 1c). The vector describing the situation, $\boldsymbol{s}$, is identical in both figures. If the player chooses to hit the ball along $\boldsymbol{s}$ as on the flat field, the ball will not be sunk. To compensate for the slope, a hitting angle larger than the one used for

[3]The shape is a scaled evaluation of the arctan function over a grid.

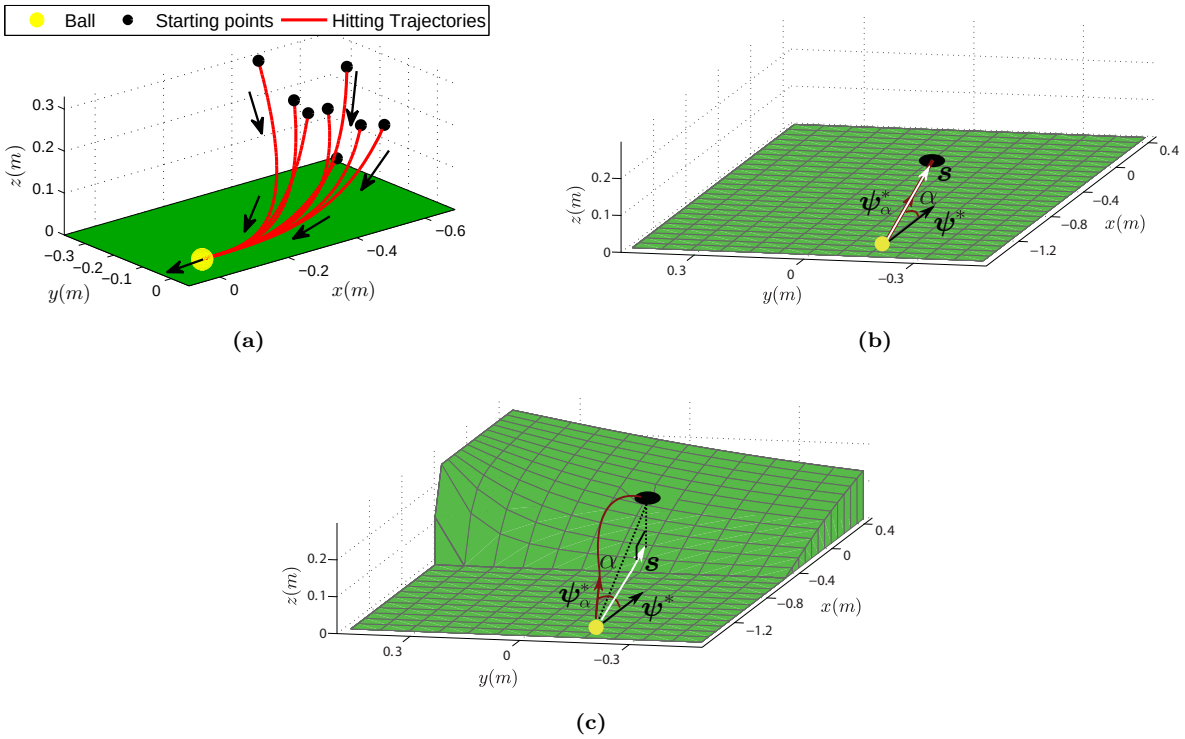


Figure 1: (a): For mastering in minigolf, the player needs to know how to swing the golf club to hit the ball in various situations. **(b) & (c):** Situation on a flat and an advanced fields. The ball trajectory of a successful attempt is indicated by the red line. For the flat field, the hitting direction should be aligned with the input vector $\boldsymbol{s}$. For the advanced field a larger hitting angle must be chosen so as to compensate for the slope of the field.

the flat field must be chosen, resulting in a curved trajectory of the ball. Changing $\boldsymbol{s}$ means that a new angle and speed must be selected accordingly. Thus, the player needs to be able to estimate the hitting angle α and hitting speed gain κ given the situation on the field $\boldsymbol{s}$. Furthermore there is generally more than one valid combination of hitting parameters for each input point on advanced fields [18]. In this paper, we refer to these different possibilities of choosing the hitting parameters as *strategies*. While learning all the strategies for a field certainly gives the player more freedom to vary her game, mastering one strategy should be sufficient for a successful game. By assuming that a strategy can be represented by a continuous mapping from the relative position of the ball and the hole to the hitting parameters, the problem is reduced to estimating this mapping:

$$\boldsymbol{g} : \boldsymbol{s} \mapsto (\alpha, \kappa) \tag{2}$$

We will address this problem in Section 5. In conclusion, the minigolf task requires two skills: 1) A default hitting motion $\boldsymbol{f}_h(\boldsymbol{x})$ that can generate motions from different initial positions and that can be altered in terms of hitting direction and hitting speed, and 2) A field-specific estimate of a mapping from input space to the hitting parameters $(\alpha, \kappa) = \boldsymbol{g}(\boldsymbol{s})$ that define what hitting parameters should be used for each situation.

4 The Hitting Motion

As outlined in the previous section, one of the requirements for the minigolf task is a default hitting motion. The hitting motion must be flexible so that the hitting direction and the hitting speed can be changed without relearning the whole motion pattern. In this section we propose a novel approach to model discrete robot hitting motions. We formalize robot motions with a *target field* and a *strength factor*. The target field defines for each point $\boldsymbol{x}$ in task space a normalized vector specifying the direction of motion, and can be viewed as a player's different techniques (e.g. topspin and slice hits in tennis). The speed of motion is defined for each point $\boldsymbol{x}$ by the scalar strength factor, and corresponds to a modulation factor to produce different desired hitting speeds:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_h(\boldsymbol{x}) = v(\boldsymbol{x})\boldsymbol{h}(\boldsymbol{x}) \tag{3}$$

where $v(\boldsymbol{x})$ denotes the *strength factor* and $\boldsymbol{h}(\boldsymbol{x})$ denotes the *target field*. Each of these terms will be described next. The structure of our formulation is inspired from many physical principles where the motion of the system in space is entirely defined by a field (e.g. gravity, electrical field, etc.) and a physical property (e.g. mass, electric charge, etc.).

4.1 The Target Field

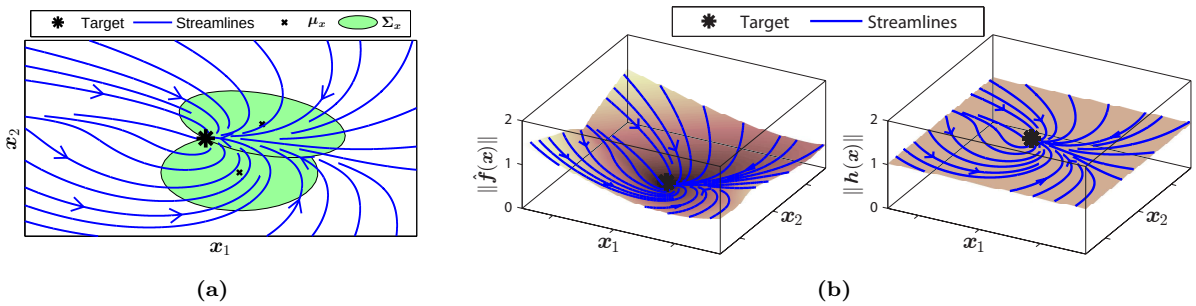
To achieve our goal of having a target field that produces trajectories that always pass through the target point with a *non-zero* velocity, we extend the original form of a globally stable time-independent DS suggested in our previous work [2]. A brief summary of this work is given below:

4.1.1 Stable Estimator of Dynamical System (SEDS) [2]

Consider a d-dimensional state variable $\boldsymbol{x} \in \mathbb{R}^d$ that can be used to unambiguously define a discrete motion of a robotic system (e.g. $\boldsymbol{x}$ could be a robot's joint angles, the position of an arm end-effector in Cartesian space, etc). Let the set of N given demonstrations $\{\boldsymbol{x}^{t,n}, \dot{\boldsymbol{x}}^{t,n}\}_{t=0,n=1}^{T^n,N}$ be instances of a global motion model governed by a first order autonomous ordinary differential equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) + \epsilon \tag{4}$$

where $\boldsymbol{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a nonlinear continuous and continuously differentiable function with a single equilibrium point $\dot{\boldsymbol{x}}^* = \boldsymbol{f}(\boldsymbol{x}^*; \boldsymbol{\theta}) = 0$, $\boldsymbol{\theta}$ is the set of parameters of $\boldsymbol{f}$, and ϵ represents zero mean Gaussian noise. In [2], we proposed a statistical based method, called *Stable Estimator of Dynamical Systems (SEDS)*, that estimates parameters $\boldsymbol{\theta}$ of a Gaussian Mixture Model (GMM) via an optimization under stability constraints. Specifically, SEDS minimizes the model estimation error given the demonstrated data while ensuring that the learned autonomous DS is globally stable at the target. The parameters $\boldsymbol{\theta}$ of a GMM are priors π^k, means $\boldsymbol{\mu}^k$ and covariance matrices $\boldsymbol{\Sigma}^k$ of $k = 1..K$ Gaussian functions (i.e.



(a) **(b)**

Figure 2: (a): Illustration of streamlines of an arbitrary 2D model learned by SEDS. **(b):** Comparison of streamlines of a SEDS model $\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$ with the target field $\boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta})$. Though both functions have exactly the same streamlines, the value of $\boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta})$ does not vanish while approaching the target.

$\boldsymbol{\theta}^k = \{\pi^k, \boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k\}$ and $\boldsymbol{\theta} = \{\boldsymbol{\theta}^1..\boldsymbol{\theta}^K\}$). The function $\hat{\boldsymbol{f}}$, the noise-free estimate of the model, is given by:

$$\dot{\boldsymbol{x}} = \hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{k=1}^{K} h^k(\boldsymbol{x};\boldsymbol{\theta})(\boldsymbol{\mu}^k_{\dot{\boldsymbol{x}}} + \boldsymbol{\Sigma}^k_{\dot{\boldsymbol{x}}\boldsymbol{x}}(\boldsymbol{\Sigma}^k_{\boldsymbol{x}})^{-1}(\boldsymbol{x} - \boldsymbol{\mu}^k_{\boldsymbol{x}})) \tag{5}$$

with

$$h^k(\boldsymbol{x};\boldsymbol{\theta}) = \frac{\pi^k \mathcal{N}(\boldsymbol{x};\boldsymbol{\theta}^k)}{\sum_{i=1}^{K} \pi^i \mathcal{N}(\boldsymbol{x};\boldsymbol{\theta}^i)} \tag{6}$$

where $\boldsymbol{\mu}^k_{\boldsymbol{x}}$, $\boldsymbol{\mu}^k_{\dot{\boldsymbol{x}}}$, $\boldsymbol{\Sigma}^k_{\boldsymbol{x}}$ and $\boldsymbol{\Sigma}^k_{\dot{\boldsymbol{x}}\boldsymbol{x}}$ are parts of the mean and the covariance of a Gaussian function $\mathcal{N}(\boldsymbol{x};\boldsymbol{\theta}^k)$:

$$\boldsymbol{\mu}^k = \begin{pmatrix} \boldsymbol{\mu}^k_{\boldsymbol{x}} \\ \boldsymbol{\mu}^k_{\dot{\boldsymbol{x}}} \end{pmatrix} \quad , \quad \boldsymbol{\Sigma}^k = \begin{pmatrix} \boldsymbol{\Sigma}^k_{\boldsymbol{x}} & \boldsymbol{\Sigma}^k_{\boldsymbol{x}\dot{\boldsymbol{x}}} \\ \boldsymbol{\Sigma}^k_{\dot{\boldsymbol{x}}\boldsymbol{x}} & \boldsymbol{\Sigma}^k_{\dot{\boldsymbol{x}}} \end{pmatrix} \quad , \quad \mathcal{N}(\boldsymbol{x};\theta^k) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}^k_{\boldsymbol{x}}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}^k_{\boldsymbol{x}})^T (\boldsymbol{\Sigma}^k_{\boldsymbol{x}})^{-1}(\boldsymbol{x}-\boldsymbol{\mu}^k_{\boldsymbol{x}})} \tag{7}$$

The resulting model from SEDS is globally asymptotically stable, i.e. starting from any point in the space, all trajectories converge to the origin. Figure 2a illustrates an example of a 2D motion constructed with $K = 2$ Gaussian functions using SEDS learning algorithm. In this model the convergence of all trajectories to the target is guaranteed, but the final velocity at the target is always zero (i.e. the target is an asymptotically stable point).

4.1.2 Target Field Formulation

To combine the stability property of SEDS with the possibility to reach the target with a non-zero velocity we define the target field as a normalized flow of motion induced by the SEDS dynamics:

$$\boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta}) = \frac{\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})}{\|\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})\|} \qquad \forall \boldsymbol{x} \in \mathbb{R}^3 \backslash \boldsymbol{x}^* \tag{8}$$

Equation (8) corresponds to a field with a constant intensity (i.e. $\|\boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta})\| = 1$), and is defined for any point in space except the target. Note that $\|\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})\| \neq 0$ everywhere except at the target point $\boldsymbol{x}^*$. The value of the target field at $\boldsymbol{x}^*$ is computed according to $\boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta}) = \lim_{\boldsymbol{x}\to\boldsymbol{x}^*} \boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta})$. The flow induced

by $\boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta})$ is of constant speed. In contrast, the SEDS flow varies according to the speed adopted during the demonstrations. The vector field $\boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta})$ conserves the convergence properties at the attractor of the SEDS flow and follows strictly the same streamlines (see Fig. 2).

Considering Eq. (8), the problem of estimating the target field $\boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta})$ is equivalent to find a globally stable DS $\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$. In this paper we propose a constrained optimization problem to compute an estimation of $\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$:

Optimization problem: Given a set of N demonstrations $\{\boldsymbol{x}^{t,n}, \dot{\boldsymbol{x}}^{t,n}\}_{t=0,n=1}^{T^n,N}$, the optimal value of the unknown parameters $\boldsymbol{\theta} = \{\pi^1..\pi^K; \boldsymbol{\mu}^1..\boldsymbol{\mu}^K; \boldsymbol{\Sigma}^1..\boldsymbol{\Sigma}^K\}$ of the function $\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$ are obtained by solving:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\sum_{n=1}^{N}\sum_{t=0}^{T^n} \omega^{t,n} \frac{(\dot{\boldsymbol{x}}^{t,n})^T \dot{\boldsymbol{x}}^{t,n}(\boldsymbol{\theta})}{\|\dot{\boldsymbol{x}}^{t,n}\|\|\dot{\boldsymbol{x}}^{t,n}(\boldsymbol{\theta})\|} \tag{9}$$

subject to

$$\begin{cases} \textbf{(a)}\ \boldsymbol{\mu}^k_{\dot{\boldsymbol{x}}} + \boldsymbol{\Sigma}^k_{\dot{\boldsymbol{x}}\boldsymbol{x}}(\boldsymbol{\Sigma}^k_{\boldsymbol{x}})^{-1}(\boldsymbol{x}^* - \boldsymbol{\mu}^k_{\boldsymbol{x}}) = 0 \\ \textbf{(b)}\ \boldsymbol{\Sigma}^k_{\dot{\boldsymbol{x}}\boldsymbol{x}}(\boldsymbol{\Sigma}^k_{\boldsymbol{x}})^{-1} + (\boldsymbol{\Sigma}^k_{\boldsymbol{x}})^{-1}(\boldsymbol{\Sigma}^k_{\dot{\boldsymbol{x}}\boldsymbol{x}})^T \prec 0 \\ \textbf{(c)}\ -\boldsymbol{\Sigma}^k \prec 0 \qquad\qquad \forall k \in 1..K \\ \textbf{(d)}\ 0 < \pi^k \leq 1 \\ \textbf{(e)}\ \sum_{k=1}^{K} \pi^k = 1 \end{cases} \tag{10}$$

where $(.)^T$ denotes the transpose, $. \prec 0$ corresponds to the negative definiteness of a matrix, and $\dot{\boldsymbol{x}}^{t,n}(\boldsymbol{\theta}) = \hat{\boldsymbol{f}}(\boldsymbol{x}^{t,n};\boldsymbol{\theta})$ are computed directly from Eq. (5). The positive weighting factors $\omega^{t,n}$ determine the relative importance of each point when computing the estimated error. In this paper, we give a lower weight ω_l and upper weight ω_u to the initial and final point of each demonstration, respectively. For intermediary points, the weighting factors are computed by linearly interpolating between these two values:

$$\omega^{t,n} = \frac{t}{T^n}(\omega_u - \omega_l) + \omega_l \tag{11}$$

Thus the optimization tries to fit the last parts of the movement better, when the effect of deviation from a desired trajectory becomes more important. Throughout this paper we use an interior-point algorithm to solve this optimization problem [24].

The optimization constraints given by Eq. (10) ensure the global asymptotic stability of the function $\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$ (see [2]). The difference between the optimization problem above and the one that is presented in [2] is in the objective function. In [2], the optimization penalizes the error in estimating both the direction and speed of movements. Considering Eq. (8), the value of $\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})$ does not affect $\boldsymbol{h}(\boldsymbol{x};\boldsymbol{\theta})$, and thus the error in estimating $\|\hat{\boldsymbol{f}}(\boldsymbol{x};\boldsymbol{\theta})\|$ should not be penalized. Equation (9), hence penalizes solely for the error in estimating the direction of movement.

4.2 Strength Factor

In order to be able to generate robot motions with similar velocity profiles as the demonstrations, we modulate the target field given by Eq. (8). The strength factor v is a *positive scalar*, and defines the intensity of a motion which the robot should follow. To capture nonlinearities in the velocity profile, we consider a varying strength factor that depends on position, i.e. $v(\boldsymbol{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$.

An estimate of the strength factor $v(\boldsymbol{x})$ can be learned using various existing regression techniques, e.g. Gaussian Process Regression (GPR) [7], Locally Weighted Projection Regression (LWPR) [25], or Gaussian Mixture Regression (GMR) [11]. In this work, we use GMR; however, one can expect similar results using the other techniques[4]. In GMR, the parameters of the Gaussian Mixture Model are optimized through Expectation Maximization (EM) [26]. EM finds an optimal model of $v(\boldsymbol{x})$ by maximizing the likelihood that the complete model represents the data well. Using GMR, the strength factor is thus given:

$$v(\boldsymbol{x}) = \sum_{k=1}^{K_{SF}} h_{SF}^k(\boldsymbol{x}) \left(\boldsymbol{\mu}_{SF,v}^k + \boldsymbol{\Sigma}_{SF,v\boldsymbol{x}}^k (\boldsymbol{\Sigma}_{SF,\boldsymbol{x}}^k)^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_{SF,\boldsymbol{x}}^k) \right) \tag{12}$$

where π_{SF}^k, $\boldsymbol{\mu}_{SF}^k$, and $\boldsymbol{\Sigma}_{SF}^k$ are priors, means and covariance of component k in the GMM model of the strength factor. The nonlinear weighting $h_{SF}^k(\boldsymbol{x})$ is computed in the same way as described by Eq. (6). The subscript SF for Strength Factor is used above to clarify that two different GMMs are involved in the reproduction of the hitting motion.

4.3 Control of Hitting Direction

Equation (3) provides the trajectory dynamics of the end-effector with the hitting speed v^* given by the strength factor at the hitting point, and the hitting direction $\boldsymbol{\psi}^*$ defined by the target field, i.e:

$$v^* = \lim_{\boldsymbol{x} \rightarrow \boldsymbol{x}^*} v(\boldsymbol{x}) \qquad \text{and} \qquad \boldsymbol{\psi}^* = \lim_{\boldsymbol{x} \rightarrow \boldsymbol{x}^*} \boldsymbol{h}(\boldsymbol{x}; \boldsymbol{\theta}) \tag{13}$$

Thus, default hitting speed and hitting direction are given during the demonstrations, which – in this work – were provided to the robot using kinesthetic teaching. To change the hitting direction and hitting speed, we proceed as follows: 1) Hitting in a different direction can be seen as a rotation of the coordinate frame in which the default DS is defined. If α is the angle between the desired and the default hitting directions in the plane of the golf field, the first step is to transform the input to the desired reference frame via the rotation matrix $\boldsymbol{R}_\alpha^T$. 2) The output of the DS needs to be transformed back to our desired hitting direction. Therefore, we rotate back through $\boldsymbol{R}_\alpha$, and 3) Finally, the hitting speed can be changed by modulating the DS by some gain κ. In brief, the following DS models a hitting motion in direction $\boldsymbol{\psi}_\alpha^*$ and with speed $\kappa v(\boldsymbol{x}^*)$:

$$\dot{\boldsymbol{x}} = \kappa \, \boldsymbol{R}_\alpha \, \boldsymbol{f}_h(R_\alpha^T \boldsymbol{x}; \boldsymbol{\theta}) = \kappa \, \boldsymbol{R}_\alpha \, v(\boldsymbol{R}_\alpha^T \boldsymbol{x}) \, \boldsymbol{h}(\boldsymbol{R}_\alpha^T \boldsymbol{x}; \boldsymbol{\theta}) \tag{14}$$

[4]Note that irrespective of which method is used, the strength factor should not affect the direction of the motion. To ensure this, the constraint $v(\boldsymbol{x}) > 0$ should be taken into account either during learning or regression.

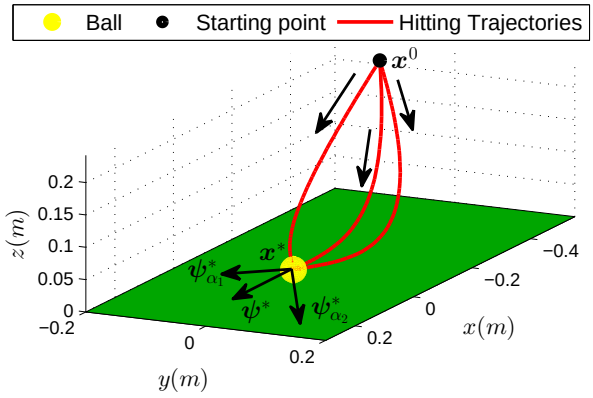


Figure 3: Control of hitting direction by using the rotation scheme. The default hitting direction $\boldsymbol{\psi}^*$ is rotated with angles α_1 and α_2 to generate hitting motions with the direction $\psi^*_{\alpha_1}$ and $\psi^*_{\alpha_2}$, respectively.

Figure 3 shows an example of using the rotation matrix to control the hitting direction at the target. In this illustration, the default hitting direction $\boldsymbol{\psi}^*$ is rotated with angles α_1 and α_2 to generate hitting motions with the direction $\psi^*_{\alpha_1}$ and $\psi^*_{\alpha_2}$, respectively.

5 The Hitting Parameters

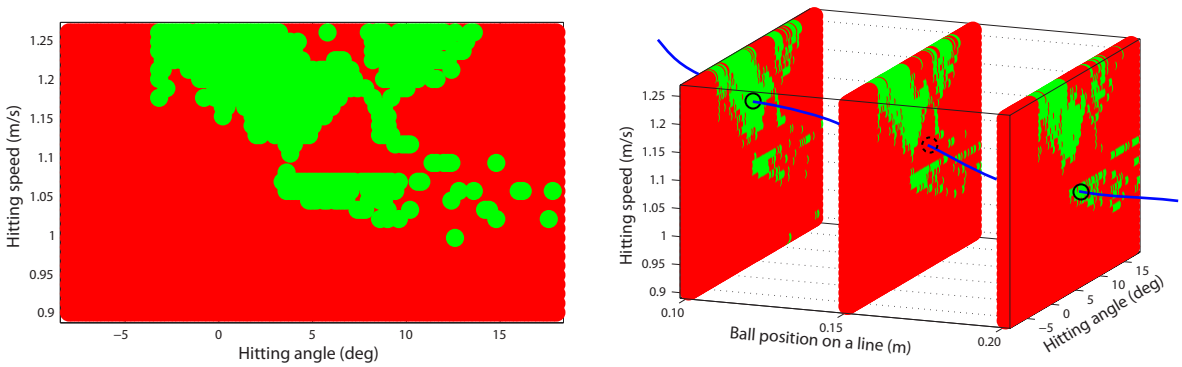
After learning an adaptable hitting motion that can be used to hit with different speed and direction, the robot needs to learn what speed and direction should be used for each situation, i.e. which κ and α should be generated for each input vector $\boldsymbol{s}$. As mentioned in Section 3, we take a supervised learning approach here and provide a training set of good parameters for different inputs. Note that the training data is field-specific, as each field requires different hitting parameters.

5.1 Training data

As mentioned in Section 3, the problem of estimating the hitting parameters based on the situation on the field is a redundant problem. There are several different strategies a player can choose from when deciding how to hit the ball. Note that within each strategy, there is a range of different angles and speeds that leads to sinking the ball, due to the fact that the hole is larger than the ball. Strategies are often represented by distinguishable separated sets of hitting parameters combinations (see Fig. 4a). Consequently, using training samples from different strategies to infer hitting parameters for new inputs will generally fail. This is illustrated in Fig. 4b. The acceptable error margins within each strategy vary in a nonlinear manner across the input space, and it is therefore not useful to determine a bound for the acceptable predictive error, as such a bound would have to be unnecessarily strict for most points to comply with the demands of the points were the acceptable error margin is small.

Consider a set of M observations of good examples[5] $\{\boldsymbol{s}^m, \alpha^m, \kappa^m\}_{m=1}^M$. Following the assumption that we are looking for a function $(\alpha, \kappa) = \boldsymbol{g}(\boldsymbol{s})$, we assume that the training set consists of noisy

[5]Note that these examples are *not* the same as the demonstrations of the default hitting motion.



(a) **(b)**

Figure 4: The left figure shows the successful (green) or unsuccessful (red) result when using the corresponding hitting parameters for a particular ball position on the arctan field. Several strategies are clearly distinguishable. The right figure illustrates the problem of picking training data from different strategies. The test point in the middle will average the two encircled training points on the left and right ball positions, resulting in the dashed encircled hitting parameters and thus failing to sink the ball.

observations of this function[6]:

$$\{\boldsymbol{s}^m, \alpha^m, \kappa^m\}_{m=1}^M = \{\boldsymbol{s}^m, g_\alpha(\boldsymbol{s}^m) + \epsilon_\alpha, g_\kappa(\boldsymbol{s}^m) + \epsilon_\kappa\}_{m=1}^M \tag{15}$$

with noise ϵ_α and ϵ_κ corrupting the angle and speed part respectively. For clarity, we introduce the following notation used specifically for the training data:

$$\{\boldsymbol{S}, \boldsymbol{\alpha}, \boldsymbol{\kappa}\} = \{\boldsymbol{s}^m, g_\alpha(\boldsymbol{s}^m) + \epsilon_\alpha, g_\kappa(\boldsymbol{s}^m) + \epsilon_\kappa\}_{m=1}^M \tag{16}$$

5.2 Hitting Parameters Prediction

In this work, we use two different statistical methods to infer the hitting parameters for new inputs using the training set specified above. In this section, we briefly review these methods. For a full derivation, refer e.g. to [7] and [27].

Consider now the mapping in Eq. (2). We assume that this mapping is drawn from a distribution over functions defined by a Gaussian Process (GP) fully specified by its covariance function. This assumption implies any set of samples from this function have a joint Gaussian distribution. By choosing the function values at the training points $\boldsymbol{S}$ with corresponding $\boldsymbol{\alpha}$ and any test point $\boldsymbol{s}^*$ and conditioning the multivariate Gaussian distribution on the training data we obtain the GPR with estimate $\hat{g}_\alpha(\boldsymbol{s}^*)$ and the predictive variance $\boldsymbol{\Sigma}_\alpha^*(\boldsymbol{s}^*)$:

$$\hat{g}_\alpha(\boldsymbol{s}^*) = \mathbf{K}_\alpha(\boldsymbol{s}^*, \boldsymbol{S})(\mathbf{K}_\alpha(\boldsymbol{S}, \boldsymbol{S}) + \sigma_n \mathbf{I})^{-1}\boldsymbol{\alpha} \tag{17a}$$

$$\boldsymbol{\Sigma}_\alpha^*(\boldsymbol{s}^*) = \mathbf{K}_\alpha(\boldsymbol{s}^*, \boldsymbol{s}^*) - \mathbf{K}_\alpha(\boldsymbol{s}^*, \boldsymbol{S})(\mathbf{K}_\alpha(\boldsymbol{S}, \boldsymbol{S}))^{-1}\mathbf{K}_\alpha(\boldsymbol{S}, \boldsymbol{s}^*) \tag{17b}$$

[6]The noise on the observations represents the small redundancies caused by the hole being larger than the ball.

The symmetric matrices $\mathbf{K}$ above represent the evaluation of the GP covariance function across the specified variables. We use a squared exponential with different length scales for the different dimensions in input space:

$$k(\boldsymbol{s}, \boldsymbol{s}') = \sigma e^{-(\boldsymbol{s}-\boldsymbol{s}')^T \mathbf{L}(\boldsymbol{s}-\boldsymbol{s}')} \qquad \text{with} \qquad \mathbf{L} = \begin{pmatrix} l_1 & 0 \\ 0 & l_2 \end{pmatrix}$$

The scalars l_1 and l_2 are the length scales of the covariance function. The scalar σ is the signal variance. We use a conjugate-gradient based search algorithm available in GPML[7] for optimizing these hyper-parameters for maximum likelihood of the training set. The above equations also apply to the hitting speed g_κ, with replacement of α and $\boldsymbol{\alpha}$ with κ and $\boldsymbol{\kappa}$ respectively[8].

Another way to infer the hitting parameters for new situations is to fit a GMM to the training set. Then, by conditioning the GMM on new query points, the corresponding hitting parameters are inferred. For a d-dimensional variable, the GMM is parameterized by $K + dK + \frac{d(d+1)}{2}K$ scalar values corresponding to the priors π^k, means $\boldsymbol{\mu}^k$ and covariance matrices $\boldsymbol{\Sigma}^k$ of the K Gaussians in the model. Given the number of Gaussian functions, or *states* in the model, the parameters can be optimized to maximize the likelihood of the training set. In this work, we first cluster the data using k-means and then apply the EM algorithm to optimize the parameters [26]. Then, GMR is used to find hitting parameters for unseen inputs:

$$\hat{\boldsymbol{g}}(\boldsymbol{s}^*) = \sum_{k=1}^{K_{HP}} h_{HP}^k(\boldsymbol{s}) \left(\boldsymbol{\mu}_{HP,\alpha\kappa}^k + \boldsymbol{\Sigma}_{HP,\alpha\kappa\boldsymbol{s}}^k (\boldsymbol{\Sigma}_{HP,\boldsymbol{s}}^k)^{-1} (\boldsymbol{s} - \boldsymbol{\mu}_{HP,\boldsymbol{s}}^k)\right) \tag{18}$$

where the nonlinear weighting $h_{HP}^k(\boldsymbol{s})$ is computed in the same way as described by Eq. (6). The subscript HP for Hitting Parameters is used above to distinguish the above GMM from those that are used in the reproduction of the hitting motion.

Note that here, we are predicting both the hitting speed and hitting angle by using a joint probability distribution over the input data and both hitting parameters. Thus, in contrast to using GPR where each parameter is predicted independently of the other, when using the GMM we take the dependency across the hitting parameters into account. Similarly, separate GMM can be built encoding the demonstrated $\{\boldsymbol{S}, \boldsymbol{\alpha}\}$ and $\{\boldsymbol{S}, \boldsymbol{\kappa}\}$ to perform GMR where the hitting parameters are predicted independently.

While GPR and GMR are both powerful methods widely used in robotics, they have some important differences in characteristics that affect how well they perform in the context of predicting hitting parameters. Consider first a flat field, as in Fig. 1b. For this field, the mapping of hitting parameters has low complexity, and a pattern observed from training data is likely valid outside the training range. As GPR is based on correlation related to the distance in input space, it outputs zero far from the training data. GMR on the other hand, has better generalization ability in that the model extends further outside the training range. Low complexity fields typically also are not very sensitive to errors in hitting parameters, i.e the precision is less important than for advanced fields. For more advanced fields such as the arctan field in Fig. 1c, higher precision is required as well as greater flexibility to capture

[7] GPML is a Matlab toolbox for GPR, written by C.E. Rasmussen and H. Nickisch.

[8] The parameters of the covariance function are also different, since these are optimized for each data set.

local patterns. GPR has better local precision than GMR, which means that it should outperform GMR for advanced fields where high precision is required when selecting the hitting parameters.

6 Minigolf Workflow

A conceptual workflow describing the learning and playing of minigolf is given in Fig. 5. The left side of this workflow explains the training parts. Three nonlinear models are learned based on two sets of user demonstrations. This procedure is performed offline. Trajectories of the hitting motions are collected through kinesthetic teaching. With these trajectories, models of the target field and the strength factor are built. These models are used to generate the hitting motion. Training data set of the hitting parameters are then collected by using the hitting motion model, with hitting parameters specified by the teacher. The teacher thus finds some examples of good hitting parameters for some situations, and adds those to the hitting parameters adaptation data set. This data set is then used as described in the previous section to estimate a mapping from the situation to the hitting parameters. Only once this hitting parameters adaptation model has been learned, the robot can play minigolf autonomously.

The right side of Fig. 5 describes the execution procedure. At each iteration, the current position of the ball, the hole, and the robot's end-effector is updated from the sensors. The correct hitting parameters are then computed using the relative position of the ball to the hole. Based on the value of the hitting angle, the rotation matrix is determined. The target field and the strength factor are then computed using the current position of the robot's end-effector and the rotation matrix. Putting together all these values, the commanded velocity to the robot is calculated using Eq. (14). This velocity is then commanded to the robot. The algorithm iterates through the steps above until it hits the ball. When the ball is hit, the motion dynamics are switched to a resting mode that smoothly stops the robot.

7 Experimental Results

As outlined in previous sections, the minigolf task requires two skills: learning the hitting motion and the hitting parameters. In order to better evaluate the performance of our system, we conducted two sets of robot experiments each of which was focused on either of the two subtasks. The first set of experiments consisted of having a 6-DoF industrial robot Katana-T arm playing minigolf on a flat field. When playing on a smooth flat field, learning the hitting parameters is unnecessary since the hitting direction is aligned with the vector connecting the centers of the ball and the hole (see Fig. 1b). The hitting speed can also be preset to a fixed value[9]. The second set of experiments focused on evaluating our system for learning hitting parameters for different challenging fields using both GMR and GPR. These experiments were performed on the 7-DoF Barrett arm manipulator. Recordings of the robot experiments can be downloaded from: `http://lasa.epfl.ch/videos/control.php`

[9]Recall that a wide range of hitting speeds can be used to sink the ball in the flat field. In this paper, we set a hitting speed of 1 m/s for the experiments on the flat field.

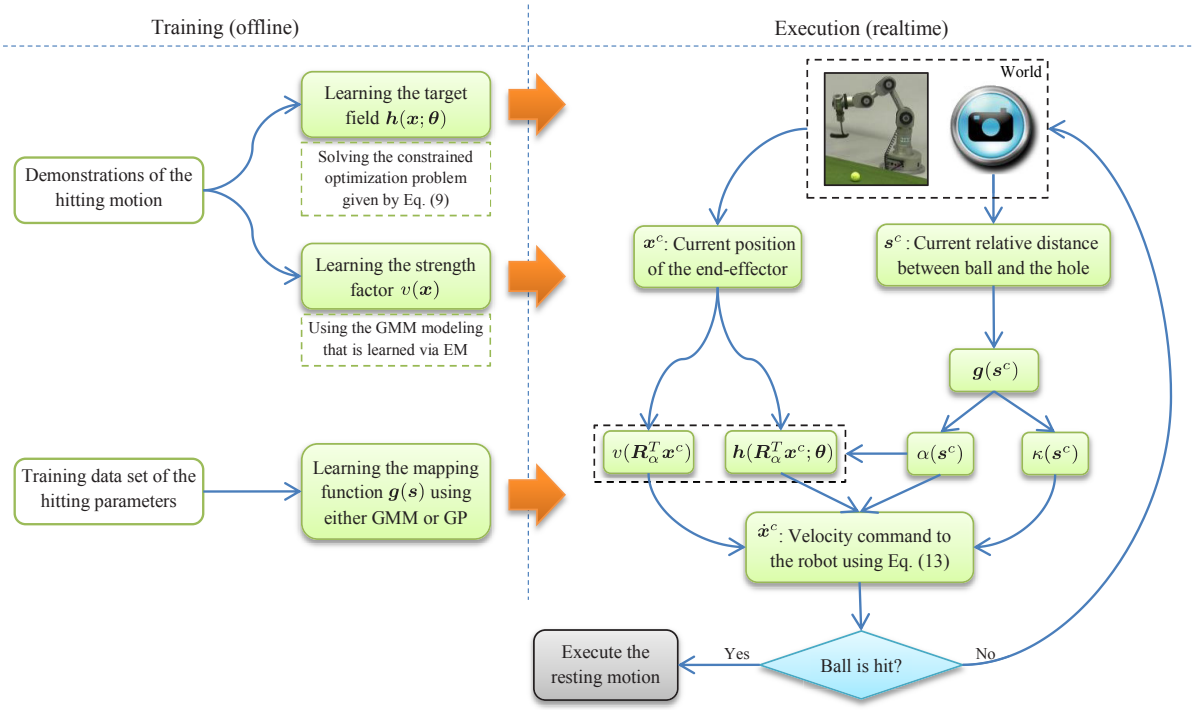


Figure 5: A conceptual workflow describing how to play minigolf. For further information please refer to Section 6.

7.1 Evaluation of Hitting Motions

The first task consisted of having a 6-DoF industrial robot Katana-T arm playing minigolf, where the robot was required to sink the golf ball into the hole on a flat field (see Fig. 6a). For this experiment, we collected a set of demonstrations by passively moving the robot arm to strike the ball. For all demonstrations, the relative position of the ball and the hole was fixed, and the user only showed the robot different ways of hitting the ball starting from different initial positions. In total, seven successful demonstrations were collected and used to learn the task (see Fig. 6b). In each demonstration, we recorded the robot's joints angles at 20Hz by directly reading them from each joint's encoder. Forward kinematics was used to compute the Cartesian position and velocity of the end-effector. This data was then used to model the task (i.e. $\boldsymbol{x} = [x\ y\ z]^T$ and $\dot{\boldsymbol{x}} = [\dot{x}\ \dot{y}\ \dot{z}]^T$). The location of the ball was detected at the rate of 80 fps using two high-speed Mikrotron MK-1311 cameras.

We solved the optimization problem presented in Section 4.1.2 to learn the target field of the hitting motion using $K = 3$ Gaussian functions. This number was selected manually based on a tradeoff between the model's accuracy and the number of parameters needed to encode the motion using the Bayesian Information Criterion. Figure 6c illustrates the reproductions of the motion using the final optimized model we obtained from the proposed extended version of SEDS. The strength factor was learned using EM algorithm with 2 Gaussian functions. Figures 6d to 6f represents the velocity profile of the reproductions versus demonstrations along the axes x, y, and z respectively.

Figure 6g shows the sequence of the motion for one of the reproductions. In our experiments, the end-effector orientation was controlled so that to keep the golf club perpendicular to the direction of

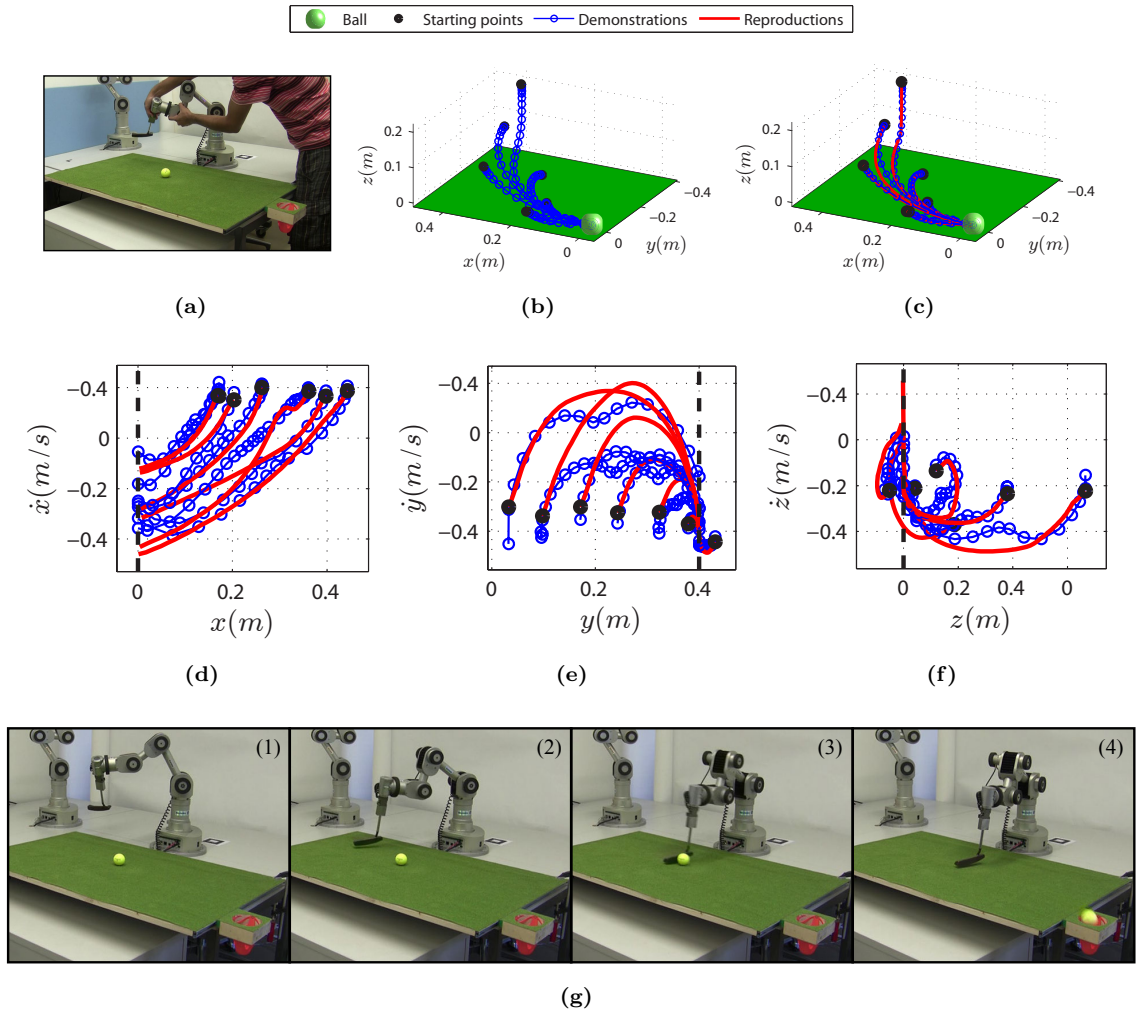
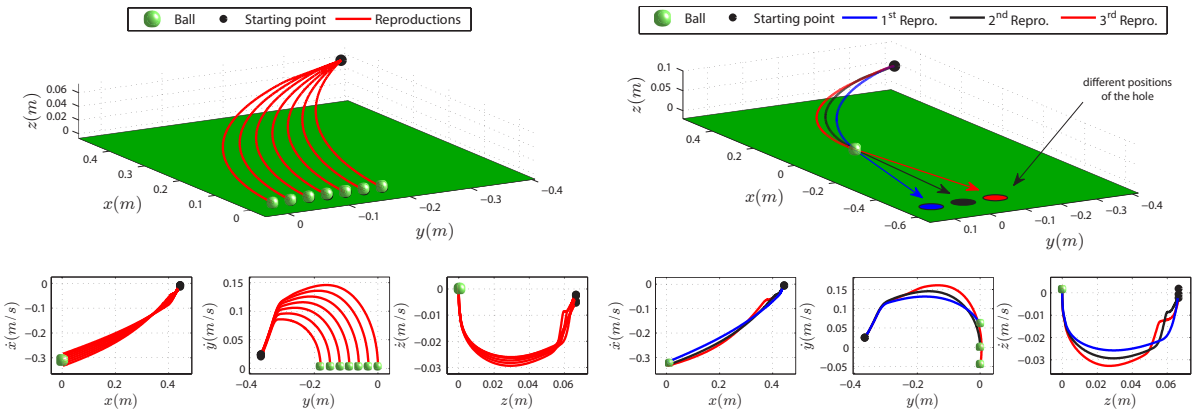


Figure 6: **(a)** Kinesthetic demonstration of putting motion to the 6-DoF Katana-T robot. **(b)** Illustration of the collected successful demonstrations. **(c)** Reproductions of the motion from the model learned with the extended version of SEDS. **(d)-(f)** Evaluation of the model's accuracy in estimating the desired velocity profile. The thick dashed lines locate the position of the ball. **(g)** Illustration of one of the generated motions sequences.

approach. At each point, the robot's joint angles were computed by solving the damped least squares pseudo-inverse kinematics. For each reproduction, after hitting the ball, the dynamics were switched to a stable dynamics guiding the arm into a resting position. For this experiment, we considered a simple resting motion where the velocity of the arm end-effector gradually decreases along the direction of the motion until it stops.

Generalization Ability: Figure 7 illustrates the generalization ability of the model to different positions of the golf ball and the hole. In Fig. 7a, we changed the position of the ball along the y axis from 0 to -0.18 m. We also changed the position of the hole so that the vector connecting the center of the ball and the hole always remained along the x axis. In all cases, the robot successfully hit the ball with the correct speed at the target. Figure 7b demonstrates the adaptation of the robot motion to three different positions of the hole. Though the initial configuration of the robot's arm and the ball positions were fixed, the robot took three different paths to hit the ball in the correct direction.



(a) Generalization to different ball positions **(b)** Generalization to different hole positions

Figure 7: Evaluation of the model's performance in generalization.

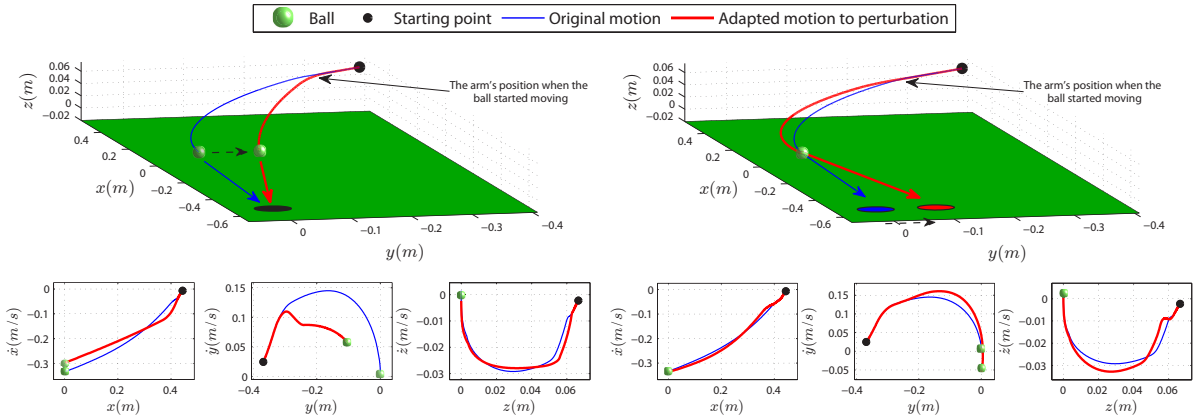
Adaptation to Perturbations: Similar to all autonomous DS, the proposed model is inherently robust to external perturbations. Figures 8a and 8b illustrate the model's behavior in the face of perturbations. In these experiments, during the robot's arm movement, the ball (Fig. 8a) and the hole (Fig. 8b) were displaced along the negative direction of the y-axis. At each time step, the robot successfully adapted its trajectory to the new position of the ball/hole until it hit the ball. In both examples, the robot successfully managed to hit the ball in the correct direction as the adaptation to the perturbation was done on-the-fly, i.e. without any re-planning. Note that despite the inherent robustness of stable autonomous DS to perturbations, there is an upper bound for the maximum value of perturbations that can be handled. This upper bound is due to the robot's torque limit, which affects the maximum acceleration the robot can achieve. Thus, if the robot faces a large perturbation when it is close to the ball, it might not be able to react swiftly and hit the ball with the correct hitting direction and speed.

7.2 Evaluation of Hitting Parameters

We evaluated the performance of our system to predict the different hitting parameters on a 7-DoF Barrett arm manipulator. The experiments on the real robot were performed on two fields: a rough flat field, and a field with two hills. The latter will be referred to as the double hill field. Model of these fields were used for experiments in a simulated environment using RobotToolKit[10], see Fig. 9. In addition to these fields, the arctan field (see Fig. 1c) was used in the simulator. Kinesthetic demonstrations from the real robot were used to learn a hitting motion model which was then used both in the simulator and on the real robot.

The minigolf playing robot uses Eq. (14) with κ and α specified either 1) By the teacher during collection of training set for hitting parameters adaptation, or 2) By the trained models presented in

[10]RobotToolKit is open-source software for simulation and real time control of robotic systems, developed by Eric Sauser at LASA, EPFL



(a) Robustness to perturbations in the ball position

(b) Robustness to perturbations in the hole position

Figure 8: Performance evaluation of the model in the face of perturbations. In this example the ball **(a)** and the hole **(b)** are pushed along the negative direction of the y-axis, as the robot approaches.

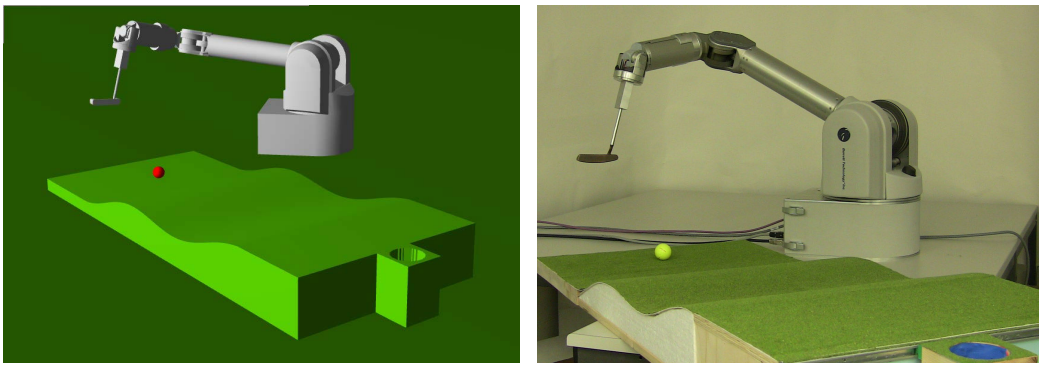


Figure 9: The double hill field in simulator (left) and with real robot (right).

Section 5 during autonomous task reproduction. In our experiments, the hitting motion was executed by first transferring the output from Eq. (14) and the end-effector orientation to joint space using the damped least squares pseudo-inverse kinematics. Then these values were converted into motor commands using an inverse dynamics controller. Both steps were carried out in realtime at 500Hz.

7.2.1 Results from the Robot Simulation

In an initial experiment, data sets consisting of 20 points were collected along one dimension in input space of the flat and the double hill fields. In practice, the input dimension was changed by moving the hole sideways along the edge of the field (see Fig. 9). The strategy was selected by fixing the speed to a constant value for all hitting attempts. A range of points around the center of the input range, represented by black crosses in Fig. 10, were selected for training. The results confirm the hypothesis that GMR has a better generalization performance outside the training set, as is clearly visible in Fig. 10.

Another experiment was centered on comparing the importance of structure when selecting training data. This is an interesting point of comparison, as the teacher might find it non-intuitive to provide training-data with some predefined structures in input-space, e.g. evenly spaced points. The data sets from the preceding experiments were used here as well. For the arctan field, a data set consisting of

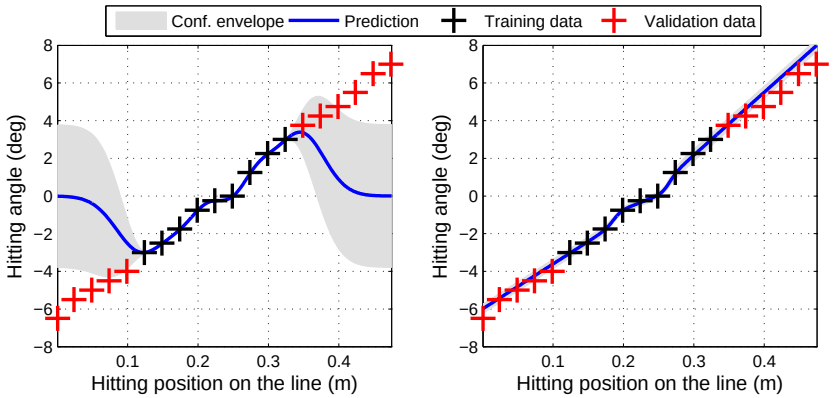


Figure 10: Red and Black crosses represent a data set of successful hitting angles for the double hill field. The points marked with black crosses were used for regression using GPR (left) and GMR (right). The gray area represents the predictive confidence by two standard deviations ($\sim$ 95 %).

Table 1: Results summary for hitting parameters learning on data from the robot simulator. In this table $\bar{e}_\alpha$, $\bar{e}_v$, %, and # refers to RMSE in angle, RMSE in speed, success rate, and number of parameters, respectively.

	Model	$\bar{e}_\alpha$	$\bar{e}_v$	%	#
The rough flat field	5 random training points				
	GPR	1.40		0.54	19
	GMR	0.52		0.64	20
	10 random training points				
	GPR	0.77		0.59	34
	GMR	0.35		0.79	20
	10 equally spaced training				
	GPR	0.15		0.96	34
	GMR	0.23		0.94	20
The Double hill field	5 random training points				
	GPR	1.60		0.36	19
	GMR	0.80		0.43	20
	10 random training points				
	GPR	1.41		0.42	34
	GMR	0.35		0.52	20
	10 equally spaced training				
	GPR	0.18		0.87	34
	GMR	0.27		0.83	20
The arctan field	16 random training points				
	GPR	0.94	0.02	0.85	72
	sep. GMR	1.01	0.02	0.86	60
	GMR	1.01	0.02	0.87	45
	28 equally spaced training				
	GPR	0.24	0.01	0.96	120
	sep. GMR	0.52	0.02	0.88	60
	GMR	0.95	0.02	0.93	45

56 points was collected. To ensure that all data points were sampled from the same strategy, we chose hitting parameters so as to minimize the hitting speed. This strategy corresponds to the lower of the three green fields representing the main strategies in Fig. 4. From the different data sets, training points were selected according to Table 1. The remainder of the data sets were used for validation of the trained models, resulting in the Root Mean Square Error (RMSE) in Table 1. The rates of success were determined by comparing random predictions for 30 datapoints selected randomly in the ranges of input used. As there are random elements both in the learning phase and more importantly in the training data selection phase, the training data selection and training were carried out 100 times for each case. The values for RMSE and rates of success are the averages of these rollouts.

The results in Table 1 clearly reveal the difference in sensitivity to the training data for the two methods. Overall GMR performs better than GPR both in terms of precision and rate of success when the training data is selected at random. However, for the evenly spaced training data, GPR clearly takes the lead. This difference is most notable for the arctan field, where the highly complex data set is handled much better by GPR. The advantage for GPR would likely be even bigger for more advanced fields. The reason the algorithms perform worse with randomly selected data is mainly because some

Table 2: Results summary for hitting parameters learning on data from the Barrett WAM robotic arm

	Model	$\bar{e}_\alpha$	%	#		Model	$\bar{e}_\alpha$	%	#
The rough flat field	5 random training points				**The Double hill field**	5 random training points			
	GPR	1.20	0.57	19		GPR	1.88	0.30	19
	GMR	0.68	0.63	20		GMR	0.72	0.37	20
	10 random training points					10 random training points			
	GPR	0.78	0.77	34		GPR	1.72	0.30	34
	GMR	0.55	0.77	20		GMR	0.42	0.40	20
	10 equally spaced training					10 equally spaced training			
	GPR	0.16	0.97	34		GPR	0.23	0.70	34
	GMR	0.22	0.90	20		GMR	0.32	0.73	20

regions in input space are likely to be poorly represented in the selected data set. Thus, there are simply no examples to learn from in these regions.

Another interesting conclusion from the results of this experiment is the higher performance of the joint GMM model versus the separate GMMs. By training one model for both hitting parameters, higher performance was achieved while using fewer parameters. In contrast to the separate GMMs, the joint GMM models the correlation between the hitting parameters. This additional information, available when training the joint GMM but not when training the separate GMMs, could possibly explain the increase in performance. The correlation is illustrated in Fig. 11. Even though we deal with very small data sets here, GMR has an advantage compared to GPR in terms of the number of parameters for all cases except when the smallest data sets are considered. Naturally, the difference in the number of parameters grows with the size of the training set.

7.2.2 Results from the Real Robot

The promising results from the robot simulator were confirmed on the real robot, using the rough flat and the double hill fields. Similarly to the simulator data sets, 20 points of successful input-parameter combinations were collected. The speed was fixed. A higher complexity was expected from these data set compared to their simulator counterparts, as a number of issues were not included in the simulator models, e.g. the dimples on the golf ball and the structure of the artificial grass covering the fields. Indeed, the data sets were more complex, which is reflected in Table 2, as the learning (with the same methods and number of Gaussians etc) yielded models poorer than those that were learned from the simulator data sets in almost all cases. When the models were trained, the hole was moved to a random location along the slider on the edge of the field, see Fig. 12. The location of the hole was captured by a stereo vision system operating at 80 fps, allowing the hitting parameters to continuously be updated to the current position of the hole. 30 points were tested to determine the rate of success[11].

[11] For the double hill we considered an upward circular shape resting motion to avoid hitting the field (as opposed to linear flat field where the velocity gradually decreases along the direction of the motion).

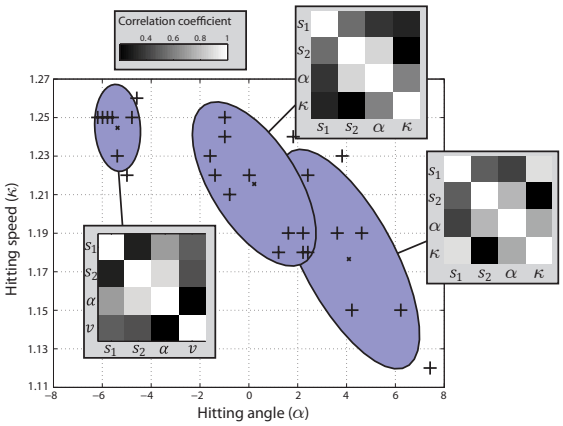


Figure 11: The figure shows the two output dimension of a GMM with 3 components fitted to a data set from the arctan field. It also illustrates the absolute correlation matrices associated to each of the Gaussians.

Figure 12: The hitting motion on the WAM. The ball and the hole are continuously tracked by a stereovision system (hence the attached red ball to the hole).

8 Conclusion and Discussion

In this paper, we showed that the complex task of playing minigolf can be learned by separating it into two subtasks 1) Learning how to hit the ball, and 2) Learning to predict the proper hitting parameters, i.e. hitting angle and hitting speed. For the first subtask, we presented a novel approach to generating robot motions with a desired velocity at the target. The new formulation has a similar structure to many physical principles, in that it computes the output of a nonlinear time-independent DS by multiplying the target field with a strength factor. For each point in space the target field indicates the correct direction of the motion, while the strength factor defines the speed of the movement in that direction. Hence it enables a robot to perform motions with similar forms but with different speeds at the target.

Similarly to a globally asymptotically stable DS, the proposed formulation is able to adapt on-the-fly a new trajectory in the face of perturbations without any need to re-index, re-scale, or re-plan. This is a critical property especially for performing agile motions. For example, in tennis, at the beginning of the motion the estimation of the ball's position is not accurate, but as the ball approaches the robot, this estimation becomes more and more accurate and thus the robot should be able to continuously adapt its motion to the new position of the ball. Note that despite the inherent robustness of stable autonomous DS to perturbations, there is an upper bound for the maximum value of perturbations that can be handled. If, for instance, the robot faces a large perturbation when it is close to the ball, due to the robot's hardware limitations the robot might not be able to react swiftly and hit the ball with the correct hitting direction and speed.

For the second subtask, we assumed that despite the many options one typically has for hitting the ball, learning one combination of hitting parameters for each input would be sufficient. In choosing this approach, the goal was to build a high performance model using only a small set of training data. These assumptions turned out reasonable, as very successful models were built from small sets of training data collected in the simulator as well as on the real robot. We showed how two different statistical methods

can be used to learn the hitting parameters selection, and compared them in terms of performance to predict hitting parameters for the task at hand. It is likely that simpler regression techniques (e.g. linear regression) could be used to predict the parameters for simple fields such as the flat field. In using a more flexible learning algorithm such as GPR or GMR, the system can handle a wider range of fields without changing the learning algorithm. Also note that by using a nonlinear regression technique, the learning of the hitting parameters can automatically compensate for errors arising from the hitting motion and/or the robot controller[12].

The proposed learning approach for the hitting parameters is able to generalize well from a small set of training data on the field for which the training data was provided. Note that the system is based on demonstrated data only, and does not use any physical model of the field. This has the advantage that the learning problem becomes relatively simple, and the disadvantage that it is not possible to generalize across different fields. A possible extension would be to reuse a basic learned model (e.g. a GMM with one or two Gaussian functions) on new fields. In such a system, the robot could exhibit very basic generalization to new fields, and the teacher could use the output from that model as a first guess when searching for successful hitting parameters.

Throughout this paper, we have highlighted the importance of choosing training data from the same strategy, as averaging samples taken from different strategies will generally lead to the selection of inappropriate hitting parameters. This high level selection of training data is intuitive to humans. Most of the previous works that deals with situation based adaptation of motions [20, 21] use reinforcement learning for learning to adapt to new situations through trial and error. Applying such an approach to hitting parameters selection in minigolf presents an interesting challenge, since the cost-function must be designed to favor only one strategy.

As mentioned, a significant simplification of the problem was made in learning only one way to hit the ball for each situation. An interesting approach would be to explore and store several successful parameters for each situation, and to cluster them into different strategies. When trained with such a data set, the robot could be programmed to use the strategy most likely to result in a successful attempt at each hitting point.

Note that the presented approach is not restricted to playing minigolf, and can be used to generate hitting motions in other tasks such as playing billiard, bowling, etc. These games may require additional hitting parameters such as spin and/or the height of release of the ball to be learned. It should be noted that our approach at its current form does not explicitly consider timely execution of the movement as it encodes hitting motions with an autonomous DS. However, in tasks where timing becomes crucial (for example in tennis), one can use an external mechanism to control the whole motion duration by actively modulating the strength factor (see e.g. the method developed in [28]). Alternatively, one could leverage on the notion of coupling across DS [29], and learn how to correctly couple the DS of the robot to that of a secondary system, describing for example the ball movement, so that both DS meet at the same location at the same time.

[12]Provided that these errors are repeatable and present during the hitting parameters demonstration phase.

Finally the proposed formulation can be used to define hitting motions in both Cartesian and Configuration (Joints) coordinate systems. In this paper we have only used the former since it was easier to work with in the context of playing minigolf. However, depending on the task at hand one can choose either of these coordinate systems.

Acknowledgment

This work was supported by the European Commission through the EU Project AMARSI (FP7-ICT-248311) and the Swiss National Science Foundation through the National Center of Competence in Research Robotics.

REFERENCES

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Handbook of Robotics*, ch. Robot Programming by Demonstration. MIT Press, 2008.

[2] S. M. Khansari-Zadeh and A. Billard, "Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models," *IEEE Trans. on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[3] E. Gribovskaya, S. M. Khansari-Zadeh, and A. Billard, "Learning Nonlinear Multivariate Dynamics of Motion in Robotic Manipulators," *The International Journal of Robotics Research*, vol. 30, pp. 1–37, 2010.

[4] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, pp. 1398–1403, 2002.

[5] M. Ramanantsoa and A. Durey, "Towards a stroke construction model," *International Journal of Table Tennis Science*, vol. 2, pp. 97–114, 1994.

[6] G. McLachlan and D. Peel, *Finite Mixture Models*. Wiley, 2000.

[7] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*. Springer, 2006.

[8] M. Matsushima, T. Hashimoto, M. Takeuchi, and F. Miyazaki, "A Learning Approach to Robotic Table Tennis," *IEEE Transactions on Robotics*, vol. 21, pp. 767–771, 2005.

[9] T. Senoo, A. Namiki, and M. Ishikawa, "Ball Control in High-speed Batting Motion using Hybrid Trajectory Generator," in *proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 1762–1767, 2006.

[10] R. Andersson, "Aggressive trajectory generator for a robot ping-pong player," *IEEE Control Systems Magazine*, vol. 9(2), pp. 15–21, 1989.

[11] S. Calinon, F. Guenter, and A. Billard, "On Learning, Representing and Generalizing a Task in a Humanoid Robot," *IEEE transactions on systems, man and cybernetics*, vol. 37, no. 2, pp. 286–298, 2007.

[12] D. Kulic, W. Takano, and Y. Nakamura, "Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains," *The Int. Journal of Robotics Research*, vol. 27, no. 7, pp. 761–784, 2008.

[13] S. Schaal, J. Peters, J. Nakanishi, and A. J. Ijspeert, "Learning movement primitives," in *Int. symposium on robotics research (ISRR2003)*, pp. 561–572, Springer, 2004.

[14] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 1293–1298, 2009.

[15] S. Calinon, E. Sauser, A. Billard, and D. Caldwell, "Evaluation of a probabilistic approach to learn and reproduce gestures by imitation," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 2671–2676, 2010.

[16] S.-M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, pp. 433–454, 2012.

[17] J. Kober, K. Mulling, O. Kromer, C. H. Lampert, B. Scholkopf, and J. Peters, "Movement Templates for Learning of Hitting and Batting," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 853–858, 2010.

[18] K. Kronander, S. M. Khansari Zadeh, and A. Billard, "Learning to Control Planar Hitting Motions in a Monigolf-like Task," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 710–717, 2011.

[19] P.-Y. Zhang and T.-S. L, "Real-Time Motion Planning for a Volleyball Robot Task Based on a Multi-Agent Technique," *Journal of Intelligent and Robotic Systems*, vol. 49, pp. 355–366, 2007.

[20] J. Kober, E. Oztop, and J. Peters, "Reinforcement Learning to adjust Robot Movements to New Situations,," in *Proc. of Robotics: Science and Systems (RSS)*, 2010.

[21] B. Nemec, M. Tamosiunaite, F. Worgotter, and A. Ude, "Task adaptation through exploration and action sequencing," in *Humanoids 2009*, pp. 610–616, 2009.

[22] P. Kormushev, S. Calinon, R. Saegusa, and G. Metta, "Learning the skill of archery by a humanoid robot iCub," in *Proc. IEEE Int. Conf. on Humanoid Robots (Humanoids)*, pp. 417–423, 2010.

[23] N. Jetchev and M. Toussaint, "Trajectory prediction: learning to map situations to robot trajectories," in *proceedings of the 26th Annual International Conference on Machine Learning*, pp. 449–456, ACM, 2009.

[24] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, "An interior algorithm for nonlinear optimization that combines line search and trust region steps," *Mathematical Programming*, vol. 107, no. 3, pp. 391–408, 2006.

[25] S. Vijayakumar and S. Schaal, "Locally Weighted Projection Regression: An O(n) algorithm for incremental real time learning in high dimensional space," in *Proc. of 17th Int. Conf. on Machine Learning (ICML)*, pp. 1079–1086, 2000.

[26] A. Dempster and N. L. D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, no. 1, pp. 1–38, 1977.

[27] C. Bishop, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[28] S. Kim, E. Gribovskaya, and A. Billard, "Learning Motion Dynamics to Catch a Moving Object," in *the 10th IEEE-RAS International Conference on Humanoid Robots*, pp. 106–111, 2010.

[29] A. Shukla and A. Billard, "Coupled dynamical system based arm-hand grasping model for learning fast adaptation strategies," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 424–440, 2012.