

Learning to Control Planar Hitting Motions in a Minigolf-like Task

Klas Kronander, Mohammad S.M. Khansari-Zadeh and Aude Billard
Learning Algorithms and Systems Laboratory (LASA)
Ecole Polytechnique Federale de Lausanne (EPFL)
{klas.kronander, mohammad.khansari, aude.billard}@epfl.ch

Abstract—A current trend in robotics is to define robot tasks using a combination of superimposed motion patterns. For maximum versatility of such motion patterns, they should be easily and efficiently adaptable for situations beyond those for which the motion was originally designed. In this work, we show how a challenging minigolf-like task can be efficiently learned by the robot using a basic hitting motion model and a task-specific adaptation of the hitting parameters: hitting speed and hitting angle. We propose an approach to learn the hitting parameters for a minigolf field using a set of provided examples. This is a non-trivial problem since the successful choice of hitting parameters generally represent a highly non-linear, multi-valued map from the situation-representation to the hitting parameters. We show that by limiting the problem to learning one combination of hitting parameters for each input, a high-performance model of the hitting parameters can be learned using only a small set of training data. We compare two statistical methods, Gaussian Process Regression (GPR) and Gaussian Mixture Regression (GMR) in the context of inferring hitting parameters for the minigolf task. We validate our approach on the 7 degrees of freedom Barrett WAM robotic arm in both a simulated and real environment.

I. INTRODUCTION

The traditional approach to controlling robots, by explicitly defining tasks by hand-coding them, is ill-suited for bringing robots in to our daily lives. Consequently, different approaches to transferring skills to robots have been proposed. One such approach is Programming by Demonstration (PbD), where tasks are demonstrated to a robot by an expert (human or robot). An important concept in PbD 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.
- 2) Given appropriate adaptation, an acquired skill can be used to carry out a more complex task than the teacher is capable of demonstrating.

In this work, we look at how a basic hitting motion model can be adapted in the context of a minigolf task. The minigolf task is a typical case covered by the motivations of adaptation presented above. A minigolf player is required to vary the hitting speed and hitting angle depending on the situation. Clearly, using the a general hitting motion model and learning how to choose the hitting parameters is more efficient than

learning a full hitting motion for each possible situation. Furthermore, a human teacher might find it difficult to actually fulfill the task, i.e. sink the ball¹, during demonstrations.

Minigolf² is a game where the players compete in sinking a golf ball in a hole on a field with various features and obstacles. The goal is to sink the ball with as few hits as possible. Depending on the features of the field, the task of estimating how to hit the ball is a difficult task that for humans takes lots of practice to master. In this work, we consider a minigolf-like task consisting in sinking the ball in one shot.

In [1], it is shown that human players in ball games usually follow the same pattern when approaching the ball, even if circumstances such as ball position change. This corresponds well to the intuition that once a player has learned to hit a ball in a particular direction and at a particular speed, she can hit in a different direction and/or with different speed using only a slightly different technique. In this work we assume that the minigolf task can be learned by mastering two subtasks that can be learned independently:

- 1) Hit the ball.
- 2) Use the correct hitting angle and speed.

For the first subtask, we consider a hitting motion modeled with Dynamical Systems (DS). We use the approach presented in [2], where demonstrations of a point-to-point motion with non-zero velocity at the target is encoded in a DS while guaranteeing global convergence at the hitting point.

The second task is learned from a training set collected with the aid of a teacher specifying good hitting parameters for some different hitting locations. We then use two statistical methods (GMR and GPR) to infer hitting parameters for unseen hitting locations.

The performance of the proposed approach is evaluated in robot experiments of playing minigolf on two different challenging fields using the 7 Degrees of freedom (DoF) Barrett WAM arm equipped with a golf club tool. Results from both environments yield a proficient mini-golf playing robot, much stronger than most inexperienced human players with the same or even higher amount of training.

¹Sinking means hitting the ball such that it goes into the hole.

²Also commonly referred to as mini-golf, miniature golf, midget golf, crazy golf and Putt-Putt

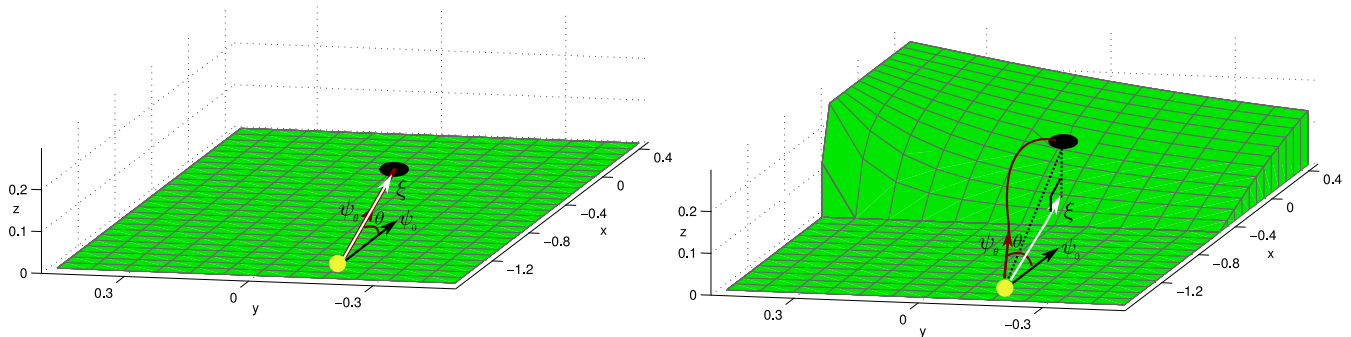


Fig. 1. Situation on a flat and an advanced field. The ball trajectory of a successful attempt is indicated by the red line. Note that ξ , the vector describing the situation (the vector from the ball to the hole projected in the hitting plane), is the same in both figures. For the flat field, the hitting direction should be aligned with the input vector, ξ . For the advanced field a larger hitting angle must be chosen, so as to compensate for the slope of the field.

II. RELATED WORK

In Programming by Demonstration (PbD), 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 learned in this way, more advanced robot motions can be achieved by combining and adapting these.

Different motion models and subsequently different learning techniques have been proposed, including but not limited to: spline-based methods [3], non-linear time dependent regression techniques [4], [5], time-dependent Dynamical Systems (DS) [6], and non-linear autonomous Dynamical Systems [7]. These methods have been successfully developed to learn motion primitives such as discrete (point-to-point) motions [7], rhythmic motions [6], hitting motions [2], [8], etc.

The focus of this paper is the adaptation of learned motion models (also commonly referred to as motion primitives). This topic has previously been studied taking a Reinforcement Learning (RL) approach in [9]. 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. Another interesting work is [10], 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. Our work differs from these in that we take a PbD approach and provide expert data to support learning. While this frees us from having to define a task-specific cost function, this assumes availability of an expert. In addition, we contrast the generalization abilities of GMR and GPR, two techniques widely used in robotics. PbD (like RL) benefit

from requiring as few demonstrations (or trials) as possible for good generalization. We aim at determining how sensitive the two techniques considered are to the amount and type of data they are provided with, contrasting in particular uniform versus sparse expert-based sampling.

III. 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 be able to hit the ball. Now assume the player has learned a planar hitting motion and can hit the ball in a direction specified by the unit vector $\psi_0 \in \mathbb{R}^2$ in the hitting plane and with hitting speed v_0 . For each new situation, a hitting angle θ and hitting speed v must be chosen such that hitting with speed v in direction $\psi_\theta = \mathbf{R}_\theta \psi_0$ (where \mathbf{R} denotes a counterclockwise rotation by θ 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 depicted in Fig. 1, left. 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 $\xi \in \mathbb{R}^2$ 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. 1 left, to play the flat field, the player simply has to align the hitting direction ψ_θ 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 in figure 1, right. The vector describing the situation, ξ , is identical in both figures. If the player chooses to hit the ball along ξ 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 the flat field must be chosen, resulting in a curved trajectory of the ball. Changing ξ means that a new angle and speed must be selected accordingly. Thus, the player needs to be able to estimate the hitting angle, θ and hitting speed, v given the situation on the field, ξ .

Furthermore there are generally more than one valid combination of hitting parameters for each input point on advanced

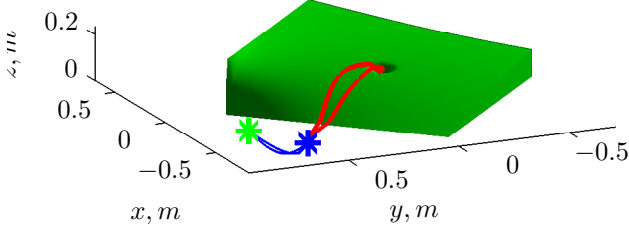


Fig. 2. The figure illustrates a typical situation for advanced fields: for a given relative position between the ball and the hole, there are several combinations of hitting speed and hitting angle that will lead to sinking the ball. The two ball trajectories are represented by the red lines. The starting point, trajectory and impact point of the end effector are represented by green star, blue line and blue star respectively. Two different strategies are applied in this figure, one with a high hitting speed and a less curved trajectory, and one where compensating for the fields slope by launching the ball at a bigger angle allows for lower hitting speed.

fields. In this paper, we refer to these different possibilities of choosing the hitting parameters as *strategies*. Fig. 2 shows samples of two strategies for one ball location for an arctan-shaped field³. 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:

$$g : \xi \mapsto (\theta, v) \quad (1)$$

In conclusion, the minigolf task requires two skills:

- 1) A default hitting motion that can be altered in terms of hitting direction and hitting speed.
- 2) A field-specific estimate of a mapping from input space to the hitting parameters $(\theta, v) = \hat{g}(\xi)$ that define what hitting parameters should be used for each situation.

In the following two sections, we give more detailed treatment to these requirements.

IV. 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. Learning a hitting motion is similar to point-to-point motions which have been extensively studied, see e.g. [7] and [6]. In this work, we use the Dynamical Systems (DS) formulation of a hitting motion, as proposed by [2]. This modeling has several advantages that make it particularly attractive in this context, the most important being:

- The hitting motion is guaranteed to converge at the hitting point from any point in space.

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

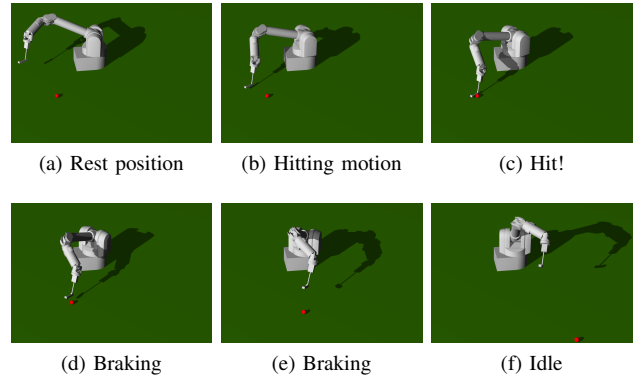


Fig. 3. Figure shows the WAM at different stages in the hitting motion. In (a), the robot is in its rest position, awaiting to initialize the hitting motion. In (b), the hitting motion has been started and the robot accelerates the end effector towards the ball. When the end effector reaches the hitting point in (c), the motion dynamics are switched to a braking mode. In (d) and (e), the robot is in the braking mode, gently lifting the golf club while smoothly decelerating the joints. When all the joints have stopped moving (f) the robot goes into idle mode.

- The time-independent DS-formulation implies inherent robustness to spatial and temporal perturbations⁴.

Fig.3 illustrates an example of the minigolf hitting motion. Here, the robot starts the motion from an initial point x^0 , and approaches the ball at position x^* along the hitting direction ψ_0 . The task space dynamics of the hitting motion (before hitting the ball) are given by the DS [2]:

$$\dot{x} = F(x) = v(x)E(x) \quad (2)$$

where $v(x)$ denotes the *strength factor* and $E(x)$ denotes the *target field*. The strength factor defines the speed of motion at each point x in task space. The target field defines for each point a normalized vector specifying the direction of motion.

In [11] it is shown how demonstrations, assumed instances of a stable DS:

$$\dot{x} = f(x) \quad \text{with} \quad f(x^*) = 0$$

can be encoded in a Gaussian Mixture Model (GMM) using the Stable Estimator of Dynamical Systems (SEDS) algorithm. In [2], this approach was generalized to cover also motions with non-zero speed at the target. Once the model has been learned, performing GMR yields an estimate \hat{f} of the DS with guaranteed stability $\hat{f}(x^*) = 0$.

$$\hat{f}(x) = \sum_{k=1}^K h_{TF}^k(x) \left(\mu_{TF,\hat{x}}^k + \Sigma_{TF,\hat{x}}^k (\Sigma_{TF,xx}^k)^{-1} (x - \mu_{TF,x}^k) \right)$$

with

$$h_{TF}^k(x) = \frac{\pi_{TF}^k \mathcal{N}(x; \mu_{TF,x}^k, \Sigma_{TF,xx}^k)}{\sum_{i=1}^K \pi_{TF}^i \mathcal{N}(x; \mu_{TF,x}^i, \Sigma_{TF,xx}^i)}$$

⁴Temporal perturbation causes the robot execution to be delayed (e.g. when slowed down because of friction in the gears) while spatial perturbations causes the robot to depart from its original trajectory (e.g. when slipping or hitting an object).

The target field $\mathbf{E}(\mathbf{x})$ is composed of unitary vectors aligned with the direction determined by the estimate $\hat{\mathbf{f}}$ and is given by:

$$\mathbf{E}(\mathbf{x}) = \frac{\hat{\mathbf{f}}(\mathbf{x})}{\|\hat{\mathbf{f}}(\mathbf{x})\|} \quad (3)$$

The speed profile is given by GMR on a different GMM, trained using the EM-algorithm [12]:

$$\mathbf{v}(\mathbf{x}) = \sum_{k=1}^K h_{SF}^k(\mathbf{x}) \left(\boldsymbol{\mu}_{SF,\dot{\mathbf{x}}}^k + \boldsymbol{\Sigma}_{SF,\dot{\mathbf{x}}}^k (\boldsymbol{\Sigma}_{SF,xx}^k)^{-1} (\mathbf{x} - \boldsymbol{\mu}_{SF,x}^k) \right)$$

which yields the strength factor:

$$v(\mathbf{x}) = \|\mathbf{v}(\mathbf{x})\| \quad (4)$$

The GMM:s are parametrized by π^k , $\boldsymbol{\mu}^k$ and $\boldsymbol{\Sigma}^k$ which represent the prior, mean and covariance of component k in the respective GMM. The indices TF for Target Field and SF for Strength Factor are used above to clarify that two different GMM:s are involved in the reproduction of the hitting motion. Note that the global convergence at the hitting point is guaranteed by the target field alone. This offers flexibility as the speed profile can be changed independently of the target field should this be desirable. In this work, we used the same demonstrations for learning both the target field and strength factor.

Eq. (2) provides the trajectory dynamics of the end effector with the hitting speed v_0 given by the Strength Factor at the hitting point, and the hitting direction $\boldsymbol{\psi}_0$ defined by the direction of approach, i.e:

$$v_0 = v(\mathbf{x}^*) \quad \text{and} \quad \boldsymbol{\psi}_0 = \lim_{\mathbf{x} \rightarrow \mathbf{x}^*} \mathbf{E}(\mathbf{x}) \quad (5)$$

Thus, a default hitting speed and hitting direction is given during the demonstrations, which are provided to the robot using kinesthetic teaching. Note that the hitting direction as defined above depends on the starting point of the hitting motion. Thus, in order to have consistent hitting direction, the hitting motion should be initiated from approximately the same point used during the demonstrations. 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 coordinates in which the default DS is defined. Thus, the first step is to transform the input to this reference, by rotating it through \mathbf{R}_θ^T .
- 2) Next, the output of the DS needs to be transformed back to our desired hitting direction. Therefore, we rotate through \mathbf{R}_θ .
- 3) Finally, the hitting speed is changed by modulating the DS by some gain s .

Concluding, the following DS models a hitting motion in direction $\boldsymbol{\psi}_\theta$ and with speed $sv(\mathbf{x}^*)$:

$$\dot{\mathbf{x}} = s\mathbf{R}_\theta \mathbf{F}(\mathbf{R}_\theta^T \mathbf{x}) = s\mathbf{R}_\theta v(\mathbf{R}_\theta^T \mathbf{x}) \mathbf{E}(\mathbf{R}_\theta^T \mathbf{x}) \quad (6)$$

The effect of the impact between the golf club and the ball depends on two things: the Cartesian trajectory of the golf club before hitting the ball (the direction of approach)

and the orientation of the golf club at the point of hitting. Human players normally align the direction of hitting with the direction of approach by keeping the golf club perpendicular to the direction of approach. In this work we take the same approach, i.e. we control the end effector orientation so as to keep it perpendicularly aligned to the hitting direction throughout the hitting motion.

V. LEARNING THE HITTING PARAMETERS

After learning an adaptable hitting motion that can be used to hit in different speed and direction, the player needs to learn what speed and direction should be used for each situation, described by the input vector $\boldsymbol{\xi}$. As mentioned, we take a supervised learning approach here and provide a training set of good parameters for some different inputs. Note that the training data is field-specific, as each different field require different hitting parameters.

A. Training data

As mentioned in section III, 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 parameter combinations, see Fig. 4 top. Consequently, using training samples from different strategies to infer hitting parameter for new inputs will generally fail. This is illustrated in Fig. 4 bottom. 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 examples⁵ $\{\boldsymbol{\xi}^m, \theta^m, v^m\}_{m=1}^M$. Following the assumption that we are looking for a function $(\theta, v) = \mathbf{g}(\boldsymbol{\xi})$, we assume that the training set consists of noisy⁶ observations of this function:

$$\{\boldsymbol{\xi}^m, \theta^m, v^m\}_{m=1}^M = \{\boldsymbol{\xi}^m, g_\theta(\boldsymbol{\xi}^m) + \epsilon_\theta, g_v(\boldsymbol{\xi}^m) + \epsilon_v\}_{m=1}^M \quad (7)$$

with noise ϵ_θ and ϵ_v corrupting the angle and speed part respectively.

For clarity, we introduce the following notation used specifically for the training data:

$$\{\boldsymbol{\Xi}, \boldsymbol{\Theta}, \mathbf{V}\} = \{\boldsymbol{\xi}^m, g_\theta(\boldsymbol{\xi}^m) + \epsilon_\theta, g_v(\boldsymbol{\xi}^m) + \epsilon_v\}_{m=1}^M \quad (8)$$

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

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

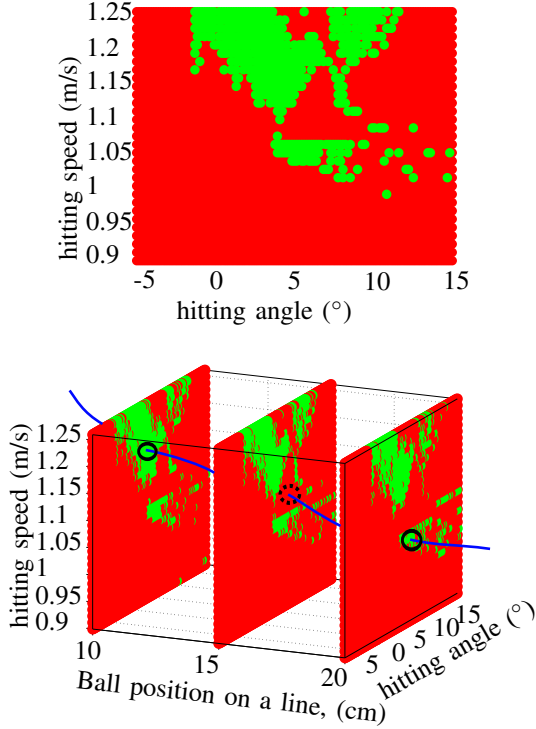


Fig. 4. The top figure shows the successful (green) or unsuccessful (red) result when using the corresponding hitting parameters for a particular ball position on the arctan field. Several strategies are clearly distinguishable. The bottom figure illustrates the problem of picking training data from 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.

B. Hitting parameter prediction using GPR and GMR

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 [13] and [14].

Consider now the mapping in Eq. (1). 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 Ξ with corresponding Θ and any test point ξ^* and conditioning the multivariate Gaussian distribution on the training data we obtain the GPR:

$$g_\theta(\xi^*) | \xi, \Theta \sim \mathcal{N}(\hat{g}_\theta(\xi^*), \Sigma_\theta^*) \quad (9a)$$

with estimate

$$\hat{g}_\theta(\xi^*) = \mathbf{K}_\theta(\xi^*, \Xi)(\mathbf{K}_\theta(\Xi, \Xi) + \sigma_n \mathbf{I})^{-1} \Theta \quad (9b)$$

and predictive variance

$$\Sigma_\theta^* = \mathbf{K}_\theta(\xi^*, \xi^*) - \mathbf{K}_\theta(\xi^*, \Xi)(\mathbf{K}_\theta(\Xi, \Xi))^{-1} \mathbf{K}_\theta(\Xi, \xi^*) \quad (9c)$$

Note that the same holds true for g_v of course, the only difference is that the θ and Θ above should be replaced by v and \mathbf{V} respectively. The \mathbf{K} -matrices above represent the evaluation of

the GP covariance function across the specified variables. We use a squared exponential with different lengthscales for the different dimensions in input space:

$$k(\xi, \xi') = \sigma e^{-(\xi - \xi')^T \mathbf{L} (\xi - \xi')}$$

where

$$\mathbf{L} = \begin{pmatrix} l_1 & 0 \\ 0 & l_2 \end{pmatrix}$$

The parameters l_1 and l_2 are the lengthscales of the covariance function. The parameter σ is the signal variance. We use a conjugate-gradient based search algorithm available in GPML⁷ for optimizing these hyper-parameters for maximum likelihood of the training set.

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. The GMM is parametrized by $K + DK + \frac{D(D+1)}{2}K$ scalar values corresponding to the priors π^k , means μ^k and covariance⁸ matrices Σ^k of the K Gaussians in the model. Given the number of Gaussians, 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 [12]. Then, GMR is used to find hitting parameters for unseen inputs:

$$\hat{g}(\xi^*) = \sum_{k=1}^K h^k(\xi^*) (\mu^k + \Sigma_{\theta v \xi}^k (\Sigma_{\xi \xi}^k)^{-1} (\xi - \mu_\xi^k))$$

with

$$h^k(\xi) = \frac{\pi^k \mathcal{N}(\xi; \mu_\xi^k, \Sigma_{\xi \xi}^k)}{\sum_{i=1}^K \pi^i \mathcal{N}(\xi; \mu_\xi^i, \Sigma_{\xi \xi}^i)}$$

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 $\{\Xi, \Theta\}$ and $\{\Xi, \mathbf{V}\}$ to perform GMR where the hitting parameters are predicted independently of each other.

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.1, left. For this field, the hitting parameter mapping has low complexity, and a pattern observed from training data is likely valid outside the training range. As the GPR is based on correlation related to the distance in input space, GPR outputs zero far from the training data. GMR on the other hand, has

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

⁸Covariance matrices are symmetric, hence the number of scalar parameters $\frac{D(D+1)}{2}$ as opposed to D^2K as the size of the matrix would otherwise imply.

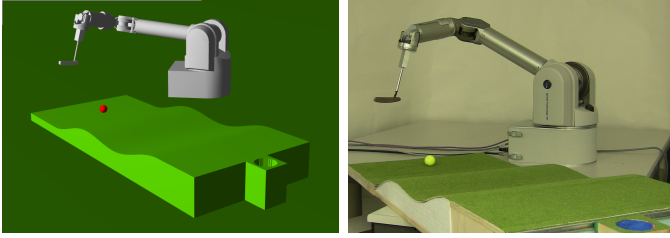


Fig. 5. The double hill field in simulator (left) and with real robot (right).

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.1, higher precision is required as well as greater flexibility to capture local patterns. GPR is much more flexible in terms of model complexity than GMR, and has far better local precision, so GPR should outperform GMR for inference within the training range for advanced fields.

VI. EXPERIMENTAL RESULTS

We tested the system and evaluated the different hitting parameter prediction methods on a 7 DoF Barrett Whole Arm Manipulator. For the experiments with the real robot two fields were used: a flat field and a field with two hills. The latter will be referred to as the double hill field. Models of these fields were used for experiments in a simulated environment using RobotToolKit⁹, see Fig. 5. In addition to these, the arctan field (see Fig. 1 and Fig. 2) 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.

A. Robot control

The minigolf playing robot uses Eq. (6) with s and θ specified either:

- 1) by a teacher during collection of training set.
- 2) by the trained models presented in section V.

The output from Eq. (6) and the end effector orientation control are then transferred to joint space using standard pseudo-inverse based Inverse Kinematics. The hitting motion is executed using torque control with a feedforward Inverse Dynamics term and a PD feedback term.

B. Hitting parameter learning in the robot simulator

In an initial experiment, data sets consisting of 20 points were collected along one dimension in input space of the flat field and double hill field respectively. In practice, the input dimension was changed by moving the hole sideways along the edge of the field (see Fig. 5). The strategy was selected by fixing the speed to a constant value for all hitting

⁹RobotToolKit is open-source software for simulation and real time control of robotic systems, developed by Eric Sauser at LASA, EPFL

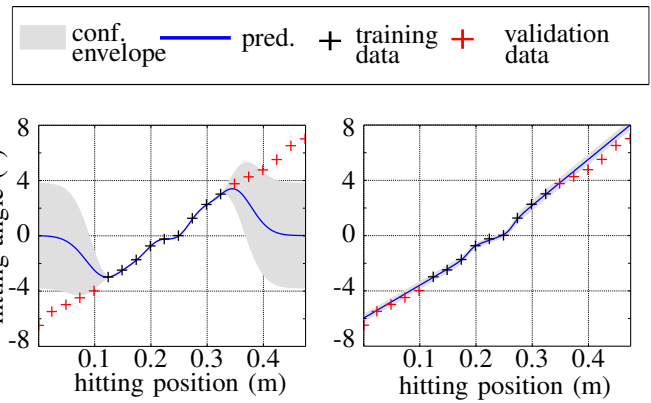


Fig. 6. Red and Black crosses represent a data set of successful hitting angles for the double hill field. The points marked with black crosses were used for regression using GPR (left) and GMR (right). The gray area represents the predictive confidence by two standard deviations ($\sim 95\%$).

attempts. A range of points around the center of the input range, represented by black crosses in Fig. 6, were selected for training. The results confirm the hypothesis that GMR has far better generalization performance, as is clearly visible in Fig. 6.

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 structure 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 56 points were 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 I. The remainder of the data sets were used for validation of the trained models, resulting in the Root Mean Square Error (RMSE) in table I. The rates of success was determined by comparing random predictions for a large number of unseen inputs with reference data sets containing all successful hitting parameter combinations for the ranges of input used¹⁰. As there are random elements both in the learning phase¹¹ 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 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

¹⁰The reference data sets were collected by programming the simulator to search the entire hitting parameter space over a fine grid in input space.

¹¹In building the GMM:s, the EM algorithm was initialized using k-means.

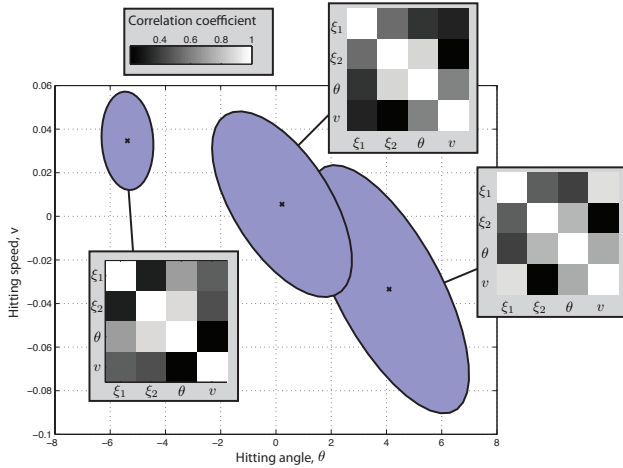


Fig. 7. The figure shows a GMM with 3 components fitted to a data set from the arctan field. The input dimension $\xi \in \mathbb{R}^2$ has been marginalized out to illustrate the correlation between the hitting speed and the hitting angle. The figure also illustrates the absolute correlation matrices associated to each of the Gaussians. As can be seen, there is very strong correlation between the hitting parameters in two of the components.

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

Another interesting conclusion from the results from this experiment is the higher performance of the joint GMM model versus the separate GMMs. By exploiting the correlation between the hitting parameters, better results are achieved by the joint GMM while using fewer parameters. The correlation is illustrated in Fig.7. Even though we deal with very small data sets here, GMR has the advantage in terms of number of parameters for all but the smallest data sets considered. Obviously, the difference in number of parameters grows with the size of the training set considered.

C. Hitting parameter learning on the real robot

The promising results from the robot simulator were confirmed on the real robot, using the flat field and the double hill field. 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 II, 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.8. The location of the hole was captured by

TABLE I
RESULTS SUMMARY FOR HITTING PARAMETER LEARNING ON DATA FROM THE ROBOT SIMULATOR

	RMSE angle	RMSE speed	Rate of success	No. of param.
The flat field				
5 random training points				
GPR	1.4021	-	0.5414	19
GMR	0.5210	-	0.6432	20
10 random training points				
GPR	0.7678	-	0.5917	34
GMR	0.3468	-	0.7889	20
10 equally spaced training				
GPR	0.1533	-	0.9564	34
GMR	0.2312	-	0.9357	20
The Double hill field				
5 random training points				
GPR	1.6021	-	0.3613	19
GMR	0.7968	-	0.4333	20
10 random training points				
GPR	1.4114	-	0.4230	34
GMR	0.3468	-	0.5170	20
10 equally spaced training				
GPR	0.1794	-	0.8721	34
GMR	0.2740	-	0.8323	20
The arctan field				
16 random training points				
GPR	0.9430	0.0200	0.8488	72
separate GMR	1.0052	0.0224	0.8568	60
GMR	1.0086	0.0229	0.8723	45
28 equally spaced training				
GPR	0.2437	0.0143	0.9643	120
separate GMR	0.5168	0.0162	0.8752	60
GMR	0.9522	0.0209	0.9255	45

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.

VII. CONCLUSION

We have shown that the complex task of playing minigolf can be learned by separating the task into hitting the ball and appropriately choosing the hitting parameters: hitting angle and hitting speed. 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

TABLE II
RESULTS SUMMARY FOR HITTING PARAMETER LEARNING ON DATA FROM
THE BARRETT WAM ROBOTIC ARM

	RMSE angle	speed	Rate of success	No. of param.
The flat field				
5 random training points				
GPR	1.2002	-	0.5667	19
GMR	0.6811	-	0.6333	20
10 random training points				
GPR	0.7849	-	0.7667	34
GMR	0.5545	-	0.7667	20
10 equally spaced training				
GPR	0.1643	-	0.9667	34
GMR	0.2226	-	0.9000	20
The Double hill field				
5 random training points				
GPR	1.8847	-	0.3000	19
GMR	0.7245	-	0.3667	20
10 random training points				
GPR	1.7179	-	0.3000	34
GMR	0.4186	-	0.4000	20
10 equally spaced training				
GPR	0.2343	-	0.7000	34
GMR	0.3194	-	0.7333	20



Fig. 8. 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).

hitting parameter selection, and compared them in terms of performance to predict hitting parameters for the task at hand.

An interesting discovery is the inherent correlation between the hitting parameters. This correlation was exploited to achieve better results with fewer parameters, using one GMM to encode both the hitting speed and the hitting angle.

One perhaps unappealing aspect of our approach is the need of a human teacher. Throughout this paper, we have highlighted the importance of choosing training data from the same strategy. While such high level selection is intuitive to

humans, it is not clear how this would be done automatically. Thus, we believe that minigolf is an example of a class of tasks that are best transferred to robots by exploiting the remarkable intuition of a human teacher. The problem of applying autonomous learning to the minigolf task lies in the cost-function on which every reinforcement learning algorithm depends. Defining such a cost-function to favor only one strategy becomes non-trivial, and might not even be possible.

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.

ACKNOWLEDGMENT

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics.

REFERENCES

- [1] M. Ramanantsoa and A. Durey, "Towards a stroke construction model," *International Journal of Table Tennis Science*, vol. 2, pp. 97–114, 1994.
- [2] S. M. Khansari-zadeh and A. Billard, "Learning to play mini-golf from human demonstrations using autonomous dynamical systems," In electronic proceeding of the workshop on New Developments in Imitation Learning, ICML, 2011.
- [3] R. Andersson, "Aggressive trajectory generator for a robot ping-pong player," *IEEE Control Systems Magazine*, vol. 9, no. 2, pp. 15–21, 1989.
- [4] 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, Part B*, vol. 37, no. 2, pp. 286–298, 2007.
- [5] D. Kulic, W. Takano, and Y. Nakamura, "Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains," *The International Journal of Robotics Research*, vol. 27, no. 7, pp. 761–784, 2008.
- [6] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "learning movement primitives," in *international symposium on robotics research (ISRR2003)*. Springer, 2004.
- [7] S. M. Khansari-zadeh and A. Billard, "Imitation learning of Globally Stable Non-Linear Point-to-Point Robot Motions using Nonlinear Programming," in *Proceeding of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.
- [8] J. Kober, K. Mulling, O. Kromer, C. Lampert, B. Scholkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *Proceeding of the International Conference on Robotics and Automation (ICRA)*, 2010.
- [9] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2010.
- [10] P. Kormushev, S. Calinon, R. Saegusa, and G. Metta, "Learning the skill of archery by a humanoid robot iCub," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, Nashville, TN, USA, December 2010, pp. 417–423.
- [11] S. M. Khansari-Zadeh and A. Billard, "Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models," *IEEE Transaction on Robotics*, 2011. [Online]. Available: <http://lasa.epfl.ch/khansari>
- [12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.
- [13] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- [14] C. Bishop, *Pattern recognition and machine learning*. Springer New York, 2006, vol. 4.