A Kinodynamic Planning-Learning Algorithm for Complex Robot Motor Control

Javier González-Quijano, Mohamed Abderrahim, Fernando Fernández, Choukri Bensalah
University Carlos III of Madrid (jgonza1,mohamed,choukri@ing.uc3m.es, ffernandez@inf.uc3m.es)

***Abstract*—Robot motor control learning is currently one of the most active research areas in robotics. Many learning techniques have been developed for relatively simple problems. However, very few of them have direct applicability in complex robotic systems without assuming prior knowledge about the task, mainly due to three facts. Firstly, they scale badly to continues and high dimensional problems. Secondly, they need too many real robot-environment interactions. Finally, they are not capable of adapting to environment or robot dynamic changes. In order to overcome these problems, we have developed a new algorithm capable of finding from scratch open-loop state-action trajectory solutions by mixing sample-based tree kinodynamic planning with dynamic model learning. Some results demonstrating the viability of this new type of approach in the cart-pole swing-up task problem are presented.**

I. INTRODUCTION

Mixing planning and model learning is indeed the core of model based learning techniques. Here, the planning process employs internal models, even if they are still immature. Every time the plan is executed, the model is updated with the new obtained experience. Later, a new planning process starts and so on. These planning-learning cycles finish when the model is good enough to generate an appropriate action sequence to reach the goal. Techniques which do not rely in prior knowledge or do not make use of an initial solution are most of the times formulated as a Markov Decision Process. Almost all algorithms solve the problem with dynamic programming related techniques. This is indeed too costly as these techniques scale very badly to continuous high dimensional problems, mainly due to their recursive nature.

The objective of this work is to develop a new algorithm based on the same ideas than model-based learning techniques, but making the planning process more efficient, even in the case of continuous high dimensional state-action spaces. In this sense, the main contribution of this work is to demonstrate that fusing randomized sample-based tree kinodynamic planners and model learning results in a new efficient way of motor control learning in complex robotic systems. The advantages provided by this algorithm should not depend on any specific randomized sample-based tree kinodynamic planner or model learning technique. Nevertheless, our developed algorithm, KiPLA (Kinodynamic Planning-Learning Algorithm), is implemented using the Blind-RRT planner in conjunction with the Local Weighted Projection Regression (LWPR) technique for the model learning process, which certainly helps on the scalability of the system.

II. RELATED WORK

Many different approaches have been developed for motor control learning problems in past years. A very nice attempt to accomplish the objective of classifying them has recently been perform by Schaal et. al. [1]. Basically, there exist three main learning schemes: model-free learning, model-based learning and mix learning approaches. Model-based approaches learn a dynamic transition model of the system at the same time that exploit it using planning. They often require less real interaction with the environment than model-free approaches as the behaviour of the system can be simulated with such models. This fact encourages the choice of this type of methods for robotics. The models are usually learned using function approximation techniques [2], [3] derived from the machine learning field. Generally, an optimal control policy, which maximizes the expected cumulative reward, may theoretically be calculated by making use of dynamic programming [4] related techniques. The control policy is generally encoded by a value function which tells the robot the expected cumulative reward from its current state while applying the optimal policy. Nonetheless, the scalability of such type of techniques to problems with high dimensional state-action spaces is still a matter of study. This is mainly due to the well known ”curse of dimensionality problem” [4]. The problem becomes even much more difficult in the case of continuous state and/or action spaces [5].

Kinodynamic planning is a relatively new field which is gaining popularity. Randomized sampling-based planners are often used to solve path-planning problems in continuous high-dimensional configuration spaces. Since LaValle published the Random Rapid Trees [6], some extensions of this popular path planning algorithm have been developed for the purpose of solving kinodynamic problems [7]. Due to the fact that kinodynamic constraints do really bias the expansion of the trees towards certain areas much more than towards others, most of this type of kinodynamic planners focus their attention in improving the state-space coverage. A recent approach regarding this issue is based on identifying less variance data distributions using PCA analysis [8]. Some other approaches are based on identifying less density areas for biasing the expansion of the tree towards those areas. In this case, the employment of grid-based data structures are the preferred choice [9], [10]. It is also possible to find some approaches for kinodynamic planning that use learning to model the environment constraints. Here, the expansion of the tree is still done on the assumption of a known dynamic model. Nonethe-

less, it is possible to combine these type of randomized sample-based kinodynamic tree planners with dynamic model learning, which may help to efficiently learn motor control policies without assuming prior knowledge. However, in the best of our knowledge, this attempt has not been done before. Next, in Section III, the KiPLA algorithm, which is precisely focused on carrying out this combination, is presented.

III. DESCRIPTION OF KIPLA

KiPLA (Kinodynamic Planning-Learning Algorithm) is indeed a new type of model-based learning technique which employs randomized sample-based tree kinodynamic planning instead of traditional dynamic programming related techniques. KiPLA begins planning with an unknown forward dynamic model. This model is used to generate plans which are not valid at the beginning of the learning process. Applying these plans enables to gather experimental data which is employed to update such model [11]. For learning the model, the Local Weighted Projection Regression (LWPR) algorithm [12], which will be explained later, is employed. This planning-learning cycle is represented in Figure 1.

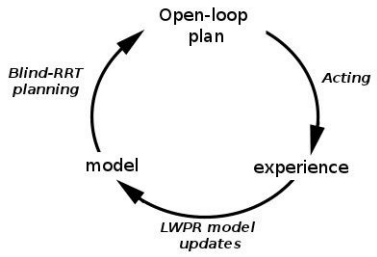

Fig. 1. Overview of the KiPLA algorithm. The planning and model learning cycle need to be repeated until convergence.

The KiPLA planning-learning cycle must be repeated until convergence. This means that the plan must offer a solution which is theoretically and in practise capable of reaching the goal. In Figure 2, some state space results corresponding to a learning episode of the cart-pole swing up task problem, before convergence has been achieved, are shown. The planning tree is represented with blue lines. At this point, the forward dynamic model is still not mature enough to provide valid solutions capable of leading the real system to the goal state. It is possible to appreciate this fact by looking at the differences between the planned state trajectory (yellow line), which ends at the goal state, with respect to the real state trajectory followed by the cart-pole system (red line).

Each time the planner is queried, in every cycle, after the model has been updated with experimental data, the obtained plans make more sense as they are constructed using all the knowledge available. After some iterations in this procedure, the planner starts providing plans which execution leads to congruent solutions that are capable of bringing the robot to the goal state.

The main advantage of KiPLA with respect to other classical model-based learning techniques is that it can quickly

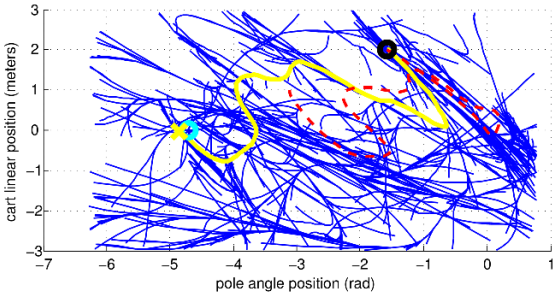

Fig. 2. Differences between the state planned trajectory and the real state trajectory during the learning process (before convergence has been achieved).

find solutions even in continues high dimensional state-action spaces. Furthermore, the LWPR is an incremental regression method which enables KiPLA to adapt to changing dynamics. The main disadvantages are that no optimal solutions are ensured and that only open-loop solutions (i.e. controller action sequences) are provided. Nevertheless, the advantages of this algorithm are good enough to consider its employment in several motor control learning problems. One of the most interesting features regarding the learning process is that the exploration is indeed the result of exploiting an immature model. The main advantage of this feature is that the trade-off between exploring and exploiting is implicitly guided.

A. Model learning through Local Weighted Projection Regression

Function approximation techniques can handle the identification of complex dynamical systems. Local approximation techniques are a preferred choice for modelling such systems. In this work, we have employed the well known Local Weighted Projection Regression technique to learn the robot system dynamics in a forward way. One of the key points of this algorithm is that it works with incremental updates, which enables KiPLA to adapt to changing environments and/or robot dynamics changes. The main difference of this algorithm with other similar ones is that it finds lower dimensional distributions of the training data and then performs the regression, thus reducing the number of local models needed to approximate the whole function. The LWPR method uses experienced data, denoted by the tuple $\{x_t, a_t, x_{t+1}\}$, as training data for the regression algorithm. The input is formed by the state and action vectors in instant time t and the output, what we want to approximate, represents the state change rate. The LWPR algorithm assumes that the objective function can be approximated using a weighted average of different linear models:

$$\hat{y}(x) = \sum_{k=1}^{K} w_k(x)\hat{y_k}(x) / \sum_{k=1}^{K} w_k(x), \tag{1}$$

where y_k are the hyperplanes representing the linear models:

$$y_k(x) = b_k^0 + b_k^T(x - c_k), \tag{2}$$

and w_k represents the kernel function which will weight the influence of each of the linear models:

$$w_k(x) = exp(-\frac{1}{2}(x - c_k)^T D_k(x - c_k)) \quad (3)$$

Given a query point, x, every linear model calculates a prediction $y_k(x)$. For nonlinear function approximation, the core concept of the learning system is to find approximations by means of piecewise linear models. Learning involves automatically determining the appropriate number of local models K, the parameters b_k of the hyperplane in each model, and also the region of validity, called receptive field (RF), parametrized as the distance metric D_k in the Gaussian kernel. Local models are then created when needed, as is deeply described in [13].

B. Blind-RRT kinodynamic planner

The Blind-RRT planner, a special kinodynamic planner we have developed for this work, takes inspiration from the kinodynamic version of RRTs [7]. It is randomized sample-based tree kinodynamic planner capable of quickly obtaining suboptimal plans, even in continues high-dimensional state-action spaces, under strong robot and environment kinodynamic constraints. The Blind-RRT uses a forward dynamics model to expand its branches. The main advantage with respect to the kinodynamic version of RRTs is that it ensures a better state-space coverage.

The way the Blind-RRT planner expands its branches is very similar to the RRT one. Both algorithms select the node that will be expanded in the same way. First, a random state is generated. Then, the nearest state node to this random state node in the tree is found. This step is the same as in the case of the classical RRT expansion. However, there is still a big difference when creating a new branch. Blind-RRT does not expand in the direction of the random state node that was used to select the expanding node. This process is described in Figure 3. A fix number of candidate actions, fulfilling the action-space constraints, are picked randomly. Then, a forward model is used to create new candidate branches (represented with discontinues lines). In the direct extension of RRTs to kinodynamic planning, the branch that most approximate to the random state should be chosen. However, due to strong biases in the candidate branches, this way of selecting the best branch does not ensure an appropriate state-space coverage. Our modification, in contrast, evaluates the final state of each of the candidate branches. The best branch is considered the one which maximizes the distance of its final state node to the nearest one in the tree. The best branch has been represented by a red discontinuous line. The distances of the final state nodes of each of the branches to its nearest neighbour (d1, d2 and d3) are represented by the ratio of the drawn circumferences, where $d1 > d2 > d3$.

IV. EXPERIMENTAL RESULTS

The cart-pole swing-up task problem is very similar to the inverted pendulum one. Both systems are classic problems in control theory. The cart-pole system is indeed an inverted

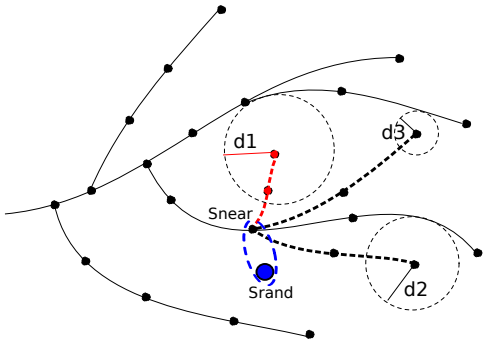

Fig. 3. Blind-RRT expands its nodes by generating random candidate branches. The one with an end state node that is more distant from any other node is finally expanded

pendulum connected with a passive joint to a cart. Unlike the problem of the cart-pole balancing task problem, the swing-up task is more difficult as it involves finding a complex policy in a large state-space volume. One of its main advantages is that it is widely used for benchmarking control and learning algorithms. The equations that have been used to simulate the dynamics of the system are described in [14]. Coulomb friction and viscous friction have also been taken into account.

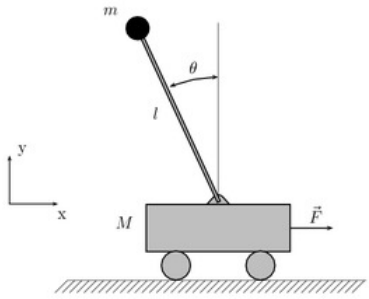

Fig. 4. Representation of cart-pole system. The joint that connects the pole with the cart is passive. The input to the system is the horizontal force applied to the cart

Relevant system variables are contained in a four dimensional state space and one action space dimension. The state vector $X(\theta, \dot{\theta}, x, \dot{x})$ represents the angular position of the pendulum, its angular velocity, the linear velocity of the cart and its linear position. The action vector is only formed by the linear force, F, exerted over the cart.

The KiPLA algorithm has been applied to the cart-pole swing-up task problem. After the learning process concludes, the real execution of the plans leads to state trajectories which match the planned ones. The state-space results, which are four dimensional, have been projected onto two-dimensional graphs. Despite the fact that there exist six possible two-dimensional projections, only four of them have been represented in Figure 5. The initial state is represented by a black circle. The complete planning tree is represented with thin blue lines. The yellow continues line represents the planned state trajectory, which is one of the branches of the tree. The red

discontinues line represents the real state trajectory followed by the cart-pole system as as result of applying the plan.

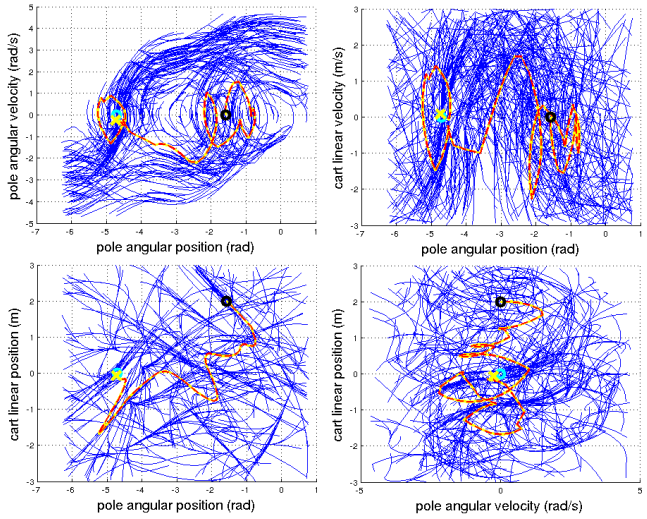

Fig. 5. State-space results of the KiPLA algorithm applied to the cart-pole swing-up tasks problem. Top-left, top-right and bottom-left graphs represents the pole angular position versus the pole angular velocity, the cart linear velocity and the cart-linear position respectively. In bottom-right graph the pole angular velocity is represented versus the cart linear position.

A graphical sequence of the cart-pole system achieving the goal state has been represented in Figure 6. Notice that the pole is first balanced to the right to get enough energy to be able to completely swing-up in the left direction. This complex policy is needed as the control signal (i.e. the force applied to the cart) was limited, not allowing to swing-up the pole directly.

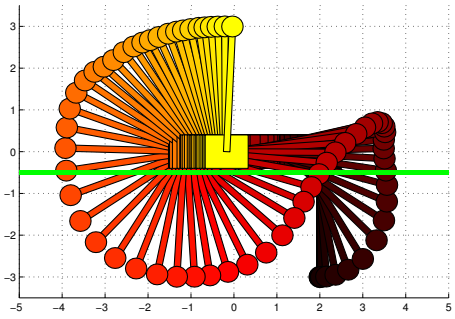

Fig. 6. KiPLA finds a compex policy to achieve goal. Balancing first to the right side is necessary for gaining enough energy to finally be able to swing-up the pole in the right direction

Due to the stochastic nature of the algorithm, the total number of learning episodes required for achieving the goal state is not constant. In the case of the experiment represented in Figure 5 and in Figure 6, the KiPLA algorithm was executed 5 times. The average number of needed learning episodes was 143. The maximum number of episodes was 235 and the minimum was 83. These results demonstrate that this new algorithm is capable of learning motor control tasks.

V. CONCLUSIONS AND FUTURE WORK

This work has proposed a new original approach to allow complex robotic systems to quickly learn new motor control policies without assuming prior knowledge about the task. The original concept of KiPLA is based on mixing randomized sample-based tree kinodynamic planning and model learning. In particular, this version of KiPLA has been implemented with our developed Blind-RRT kinodynamic planner and the Local Weighted Projection Regression Algorithm. The Blind-RRT algorithm allows to efficiently look for solutions in continuous high dimensional state-action spaces. The employment of the LWPR algorithm for the model learning process also enables KiPLA to scale well to these kind of state-action spaces. Future work will be focused in providing KiPLA with active learning capabilities based on probabilistic plans. This will help to decide which action sequence should be chosen in order to improve the model in the regions of interest using less trials. In addition, the probabilistic may also help to control the risk of executing certain actions, thus allowing for safety exploration.

VI. ACKNOWLEDGMENTS

The research leading to these results has been partially supported by the HANDLE project, which has received funding from the European Communitys Seventh Framework Program (FP7/2007-2013) under grant agreement ICT 231640

REFERENCES

[1] S. Schaal and C. G. Atkeson, "Learning control in robotics," *IEEE Robotics and Automation Magazine*, 2012.
[2] *Modern Multivariate Statistical Techniques: Regression, Classication, and Manifold Learning*. Springer–Verlag, 2008.
[3] D. Nguyen-tuong, "Model learning in robot control," Ph.D. dissertation, University of Freiburg, 2011.
[4] *Dynamic programming*. Princeton University Press, 1957.
[5] J. C. Santamaria, R. S. Sutton, and A. Ram, "Experiments with reinforcement learning in problems with continuous state and action spaces," *Adaptive Behaviour*, 1998.
[6] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.
[7] S. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *IEEE International Conference on Robotics and Automation*, vol. 1, 1999, pp. 473–479.
[8] Y. B. K. E. Li, "Balancing state-space coverage in planning with dynamics," in *International Conference in Robotics and Automation*. MIT Press, 2010, pp. 233–241.
[9] I. A. Sucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Workshop on the Algorithmic Foundations of Robotics*.
[10] A. M. Ladd and L. E. Kavraki, "Motion planning in the presence of drift, underactuation and discrete system changes," in *Robotics: Science and Systems*. MIT Press, 2005, pp. 233–241.
[11] *From Motor Learning to Interaction Learning in Robots*. Springer-Verlag, 2010, ch. Learning Forward Models for the Operational Space Control of Redundant Robots.
[12] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space," in *In Proceedings of the Seventeenth International Conference on Machine Learning (ICML*, 2000, pp. 1079–1086.
[13] S. Vijayakumar, D. A., and S. Schaal, "Lwpr: A scalable method for incremental online learning in high dimensions," Edinburgh University Press, Tech. Rep., 2005.
[14] R. V. Florian, "Correct equations for the dynamics of the cart-pole system," Center for Cognitive and Neural Studies, Tech. Rep., 2007.