Towards Non-Parametric Bayesian Learning of Robot Behaviors from Demonstration

Stéphane Magnenat
Autonomous Systems Lab, ETH Zürich
Tannenstrasse 3, 8092 Zürich, Switzerland
stephane at magnenat dot net

Cédric Pradalier
Autonomous Systems Lab, ETH Zürich
Tannenstrasse 3, 8092 Zürich, Switzerland
cedric.pradalier@mavt.ethz.ch

Francis Colas
Autonomous Systems Lab, ETH Zürich
Tannenstrasse 3, 8092 Zürich, Switzerland
francis.colas@mavt.ethz.ch

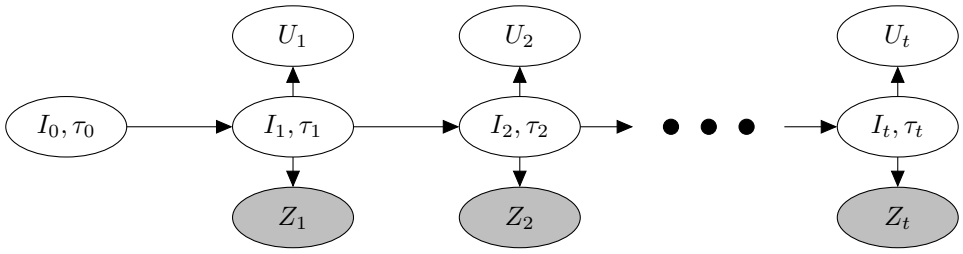
1 Introduction

A large variety of mobile robots has blossomed recently, made possible by progresses in energy storage, electronics, processing speed, etc. We have seen these robots performing exploration and navigation tasks in different environments, and demonstrating impressive autonomy. However, the actual programming of complex behaviors, in particular when physical interaction takes place between the robot and its environment, has remained tedious. This work often involves writing and tuning complicated state machines and requires a lot of engineering resources. Machine learning, which aims at specifying complex algorithms by leveraging data instead of just expert knowledge, proposes an alternative path. It appears especially useful for robotic tasks, as these often deal with high-dimensional input and output spaces while relying on a lot of contingent parameter values.

Since the early turtles from Grey Walter whose behaviors were defined by simple analog circuits, many approaches have been developed to specify robot behaviors from recorded training data, forming the field of *programming by demonstration*. While this field ultimately aims at finding a solution to program any robotic task, currently the choice of the approach mostly depends on the kind of tasks to be executed. Indeed, for robotic arms that can have many degrees of freedom and are required to follow precise trajectories with continuity constraints, regression models have been proposed to extract a functional representation of a trajectory from training data. For example, Calinon et al. [1, 2] use Gaussian models to do regression and generalization between several recorded trajectories. They focus on the replication of a motion primitive and do not tackle a discrete sequence of steps in a task.

On the other hand, several approaches split training sequences into motion primitives by specifying either a structure for the task [3] or a function deciding on a step change, typically looking for discontinuities [4, 5, 6] or using ad-hoc information like contact points [7]. These approaches can handle robotic tasks involving several steps but require a definition, either explicit or implicit, of those steps. As a consequence, they need a lot of parameters that depend both on the robotic system and on the task. Recent work has tackled the question of reducing this number of parameters, for instance by trying to learn both transitions and states in an unsupervised way [8]. However, the latter study concludes that this is not yet possible.

Our aim is to understand the limits of capabilities that can be achieved with a simple, non-parametric system that learns from demonstration. We want to reduce the number of meta parameters to the minimum, because these ultimately have to be given to the system, while still handling tasks consisting of several steps. This paper proposes a system with only $n_s + 2$ meta parameters, where n_s is the number of sensor dimensions. All these parameters are related to the robotic platform and its application, but do not depend on the task. We believe that such a system, should it perform well



$$
\begin{aligned}
&p(U_{1:t}, Z_{1:t}, I_{1:t}, \tau_{1:t}) = \\
&p(U_{1:t-1}, Z_{1:t-1}, I_{1:t-1}, \tau_{1:t-1}) p(U_t|I_t, \tau_t) p(Z_t|I_t, \tau_t) p(I_t|I_{t-1}) p(\tau_t|\tau_{t-1})
\end{aligned} \tag{1}
$$

Figure 1: The graphical representation of the model and the corresponding decomposition.

enough, would be of tremendous help for developing and deploying robotic applications. Indeed, the current requirement of choosing by hand many parameters is a major obstacle to the deployment of programming by demonstration.

2 Model

Our approach does not try to build a synthetic representation of the training data. On the contrary, the algorithm aims at tracking, in the training data, the most relevant information with respect to the current attempt to reproduce the behavior. More precisely, we build a Bayesian filter in which we assume that both sensor readings and motor commands are conditioned by trajectory and time indices. In this model, replaying is done by inferring the motor commands by marginalization over those trajectory and time indices. The model also assumes that motor commands and sensor observations are available at each time step at a certain constant frequency. This model builds on the work of Pradalier and Bessière [9], adding multiple trajectories.

2.1 Variables

The formal expression of this model involves defining several variables:

- $\Pi = \{\zeta_t^i, \upsilon_t^i | \forall i \in (1, N), \forall t \in (1, L_i)\}$ Records of N trajectories of lengths $\{L_1, L_2, \ldots, L_N\}$, where trajectory i, at record time step t, has sensor data ζ_t^i and actuator command υ_t^i (vector values). All subsequent formulas are assumed to be conditioned by Π.
- I_t Index of trajectory at replay time t, ranges from 1 to N.
- τ_t Position on trajectory at replay time t, ranges from 1 to $\max_i L_i$.
- U_t Actuator command at replay time t, vector value.
- Z_t Observation (sensor data) at replay time t, vector value of n_s dimension.

2.2 Distributions

We decompose the joint distribution over those variables by leveraging independence assumptions. This leads to a recursive expression of the inference similar to a Bayesian filter (see Figure 1).

The distributions involved are:

- $p(U_t|I_t, \tau_t)$: not used in the inference directly, see Section 2.3.2,
- $p(Z_t|I_t, \tau_t)$: an observation model,
- $p(I_t|I_{t-1})$: transition between trajectories,
- $p(\tau_t|\tau_{t-1})$: transition from a time step to the next,
- $p(I_{t-1}, \tau_{t-1}|Z_{1:t-1})$: result of the previous step of inference.

2.2.1 Parameters

Our model has only $n_s + 2$ meta parameters: θ_I and θ_τ that control transitions, plus a single parameter σ_{ζ^k} for each sensor dimension k. This parameter is related to the scale of variation: it basically states when two values are different. It is important to note that this parameter is only lower-bounded by the noise of the sensor but is mostly governed by the semantics of the values. This means that for two sensors adequately measuring the same quantity, this value would be the same even if one sensor is less noisy than the other. For example, for an outdoor wheeled robot, both differential GPS and odometry give an information on the position, but with a different precision. In that case, if the task is just to reach a large area, the scale factor can be in the order of the meter for both sensors, even if the d-GPS can achieve better precision. Hence while the σ_{ζ^k} parameters might vary for different application contexts, they would be similar for different tasks within the same context, like reaching different areas in the preceding example.

2.2.2 Observation model

The observation model is not trivial, in particular because in some runs, two successive data points can be as far away as 90 % of the space. Therefore, a simple Gaussian modeling sensor noise situated on the data points is not enough, as it would lead to infinitesimal probabilities when two successive data points are very far away. To cope with this, we consider that the observations are sampled from a piecewise linear function. Thus, we convolve a Gaussian with both segments in observation space linking observations at time $t-1$ and t, and t and $t+1$:

$$p(Z_t|I_t = i, \tau_t = j) = \prod_k \left[\int_{\zeta^i_{j-1}}^{\zeta^i_j} \frac{1}{2(\zeta^i_j - \zeta^i_{j-1})_k} \mathcal{N}(t, \sigma^2_{\zeta^k}) dt + \int_{\zeta^i_j}^{\zeta^i_{j+1}} \frac{1}{2(\zeta^i_{j+1} - \zeta^i_j)_k} \mathcal{N}(t, \sigma^2_{\zeta^k}) dt \right] \tag{2}$$

This can be implemented efficiently using the `erf` function.

2.2.3 Transition model

The transition model is structured in two parts: the index of the trajectory and the time position in a given trajectory. The distribution over the next trajectory index given the past trajectory index is close to an identity matrix but with uniform probability θ_I to jump from one trajectory to another. This ensures a strictly positive lower bound on the probability of each trajectory, which is useful to allow a trajectory that differs from the first part of the observations to meaningfully contribute to the motor commands when the observations start matching again:

$$p(I_t|I_{t-1}) = \begin{cases} 1 - \theta_I & \text{if } I_t = I_{t-1} \\ \frac{\theta_I}{N-1} & \text{otherwise} \end{cases} \tag{3}$$

The transition for the position inside a given trajectory expresses that this position index most likely gets increased by 1 but can also increase by 2 or not increase at all, with a probability θ_τ. This allows for slight extensions or compressions of time for the replay, according to the observations:

$$p(\tau_t|\tau_{t-1}) = \begin{cases} \theta_\tau & \text{if } \tau_t = \tau_{t-1} \\ 1 - 2\theta_\tau & \text{if } \tau_t = \tau_{t-1} + 1 \\ \theta_\tau & \text{if } \tau_t = \tau_{t-1} + 2 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

2.2.4 Initial conditions

The initial condition is a uniform distribution over the trajectories and a Dirac delta function on the first time step:

$$p(I_0 = i, \tau_0 = j) = \begin{cases} 1/N \text{ if } j = 0 \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

2.2.5 Termination criterion

The task is considered completed if $p(\tau_t \text{ in last 10 time steps}) > 0.9$.

2.3 Questions

The inference can be divided into three different questions:

- update due to time, involving the prediction model;
- generation of a command, involving a decision function;
- update due to observations, involving the observation model.

This order is chosen to allow commands to be triggered by time rather than by observation. Indeed, if a change in observation depends on a specific action, the replay needs to actually make this action in order for the sequence to move on.

2.3.1 Prediction update

The prediction update applies the transition models for time and trajectory indices. It corresponds to the following inference:

$$\begin{aligned} p(I_t, \tau_t | Z_{1:t-1}) &= \sum_{I_{t-1}, \tau_{t-1}} p(I_t, \tau_t | I_{t-1}, \tau_{t-1}) p(I_{t-1}, \tau_{t-1} | Z_{1:t-1}) \\ &= \sum_{I_{t-1}} p(I_t | I_{t-1}) p(I_{t-1} | Z_{1:t-1}) \sum_{\tau_{t-1}} p(\tau_t | \tau_{t-1}) p(\tau_{t-1} | I_{t-1}, Z_{1:t-1}) \end{aligned} \tag{6}$$

2.3.2 Getting command U_t at time t

The probability of a given command U_t is the marginalization over trajectory and time indices:

$$p(U_t | Z_{1:t-1}) = \sum_{I_t, \tau_t} p(U_t | I_t, \tau_t) p(I_t, \tau_t | Z_{1:t-1}) \tag{7}$$

This expression depends on $p(U_t | I_t, \tau_t)$, which is not known, and of the prediction update computed above. However, if we assume that the trajectories have been generated by applying a decision function D on the probability distribution over the commands, and that this function is linear, we do not need to specify $p(U_t | I_t, \tau_t)$. Indeed, distributing D in Equation 7 yields:

$$\begin{aligned} D(p(U_t | Z_{1:t-1})) &= \sum_{I_t, \tau_t} D(p(U_t | I_t, \tau_t)) p(I_t, \tau_t | Z_{1:t-1}) \\ &= \sum_{I_t, \tau_t} \upsilon_{\tau_t}^{I_t} p(I_t, \tau_t | Z_{1:t-1}) \end{aligned} \tag{8}$$

where we can assume $D(p(U_t | I_t = i, \tau_t = j)) = \upsilon_j^i$. In the end, the command is a linear combination of commands from the reference trajectories.

2.3.3 Taking sensor data into account

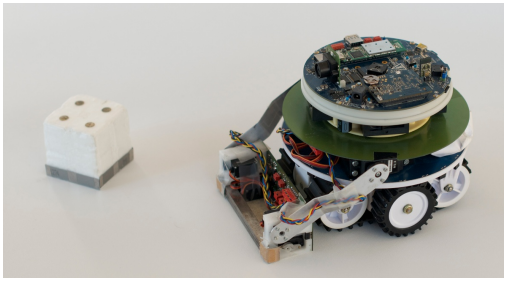
The internal state is then finally updated using the observation:

$$p(I_t, \tau_t | Z_{1:t}) \propto p(Z_t | I_t, \tau_t) p(I_t, \tau_t | Z_{1:t-1}) \tag{9}$$

3 Experiments

This experiment consists in grasping a polystyrene cube with a miniature mobile robot [10], equipped with a magnetic gripper (Figure 2, left). The robot has 5 degrees of freedom: its two tracks, the elevation and tilt angles of its gripper, and its on/off switch [11]. The sensors consist of a camera and six infrared-based proximeters. The camera is pre-processed to return the position of the cube on the x-axis in the image. The proximeters are pre-processed to return a linearized value.

The cube is placed at a distance of 25–40 cm of the robot, with a horizontal shift of $\pm$ 10 cm (Figure 2, right). The robot must orient towards the cube, change the position of its gripper to scan the cube, advance until it is close enough, refine its orientation, turn its gripper back in the grasping



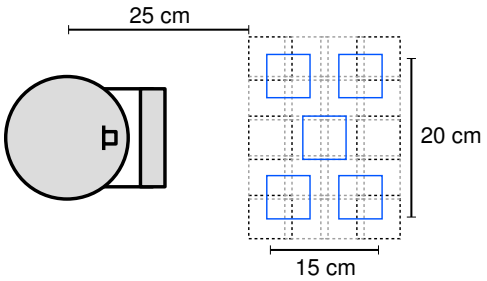


Figure 2: The robot with the cube (left) and the experimental setup (right).

with 20 demonstrations:				
θ_I	S	Fs	Fg	Fb
1e-0.5	14	1	0	0
1e-1	13	1	0	1
1e-1.5	13	0	1	1
1e-2	14	0	1	1
1e-2.5	7	6	1	1
1e-3	11	2	2	0
1e-4	9	5	0	1
with 6 demonstrations:				
1e-0.5	11	3	1	0

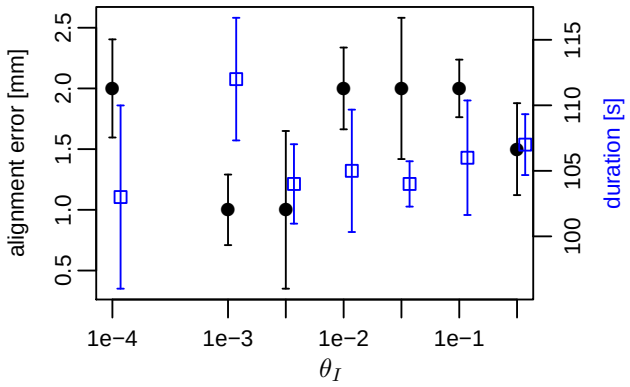


Figure 3: Experimental results for different θ_I. Left, the outcomes of the runs: S means success, Fs means that experiment duration exceeded 3 minutes, Fg means that the robot failed to grasp the cube and Fb means that it tried to exit the experimental area. Right, the alignment error and run duration: points are averages and bars are standard errors.

position, and then fetch the cube. We choose this problem because programming and tuning this behavior has required a significant effort in a previous work [12]. We use this scenario to validate the model on a safe system, to study the influence of the parameter θ_I, and to test the interpolation capabilities.

To study the influence of the θ_I, we recorded 20 training runs with the cube at 20 different positions (dashed squares in Figure 2). Then, for 7 different values of θ_I, we tested 3 times the fetching of the cube at 5 different positions, that were not present in the training data (blue squares in Figure 2). The parameter θ_τ is fixed at 0.05. For a value of θ_I comprised between 1e-0.5 and 1e-2, the success rate is high at about 90 % (Figure 3, left). For smaller values, it drops because at some point, the robot either does not move any more or performs a movement repeatedly. When runs are successful, the alignment error is always small and the duration relatively constant (Figure 3, right). We believe that a large θ_I leads to the best performances because our training runs essentially differ at the beginning, and therefore a large θ_I allows more possibility to fine tune the behavior afterwards, by jumping to a different trajectory that fits better the observation.

Most of the failures (60 %) are linked to the controller stopping the robot indefinitely, due to a fixed or cyclic distribution on I_t, τ_t. We attribute this effect to two causes. First, motor commands are discretized with a relatively low resolution, preventing the robot from moving if, for instance, the motor command is 0.4. We could alleviate this problem through temporal dithering, by probabilistically selecting the nearest integers in proportion to their distance. The second problem is due to the servo motors that actuate the gripper. As they are controlled in position, they do not cope well with rapid changes of set points. We have observed that they perform the best when the battery is fully charged, because their speed is directly proportional to the battery voltage. The other types of failure are linked to the robot missing the cube or exiting the experimentation area.

To test the interpolation capabilities, we have used only 6 runs out of 20 (dashed black squares in Figure 2) and applied the same test procedure as before, with θ_I fixed to 1e-0.5. Out of 15 test runs,

11 were successful, 3 failed because the controller stopped indefinitely, and 1 failed because the robot did not align properly with the cube. In the successful runs, the mean error was 2.1 mm and the mean duration was 99.5 s, which is similar to the results with 20 training runs. This shows that our model is able to interpolate between training data and that its performances degrade gracefully when less data are available.

4 Discussion

Given that N is smaller than L_i, and considering that $p(\tau_t|\tau_{t-1})$ is zero excepted when τ_t and τ_{t-1} are adjacent or equal, Equations 7 and 9 have a similar complexity of $O(L \times N)$. This complexity is tractable on current laptops, for dozens of trajectories and thousands of time steps.

Because our model takes little a-priori knowledge, its generalization ability is limited. Given meaningful σ_{ζ^k}, it can interpolate but not extrapolate. It can, however, use parts of different trajectories during execution. We believe that in many practical scenarios, this is sufficient because extrapolation is not needed or desired. Note that the sensor space should be isotropic, otherwise σ_{ζ^k} has little meaning.

The main theoretical limitation of our model is the lack of abstraction. First, it assumes that no sensor variable is independent of the current action of the robot, as otherwise, because of the low number of demonstrated trajectories, such a variable would disturb the replay. This is a strong limitation, and albeit the application developer could select the variables to give to the model, as we do for the camera, this solution does not fit our philosophy of having as few parameters as possible. Moreover, in some applications [13], the relevant variables change in the course of the trajectory. In the future, we will explore how to identify those that support motor commands, in the direction of [7, 4]. Second, our model is not explicitly able to handle loops, although practically it might be (up to a certain point) due to the non-zero probability of past time steps. We believe that on the long run, trajectory parts should be abstracted and put into relation. This is a difficult problem, and a compression-based approach might be a good research direction.

On the short term, future work includes testing this model on different platforms and comparing its performances with related work on similar tasks. The current lack of common platform/test scenario is an obstacle, that could be alleviated through joint studies.

5 Conclusion

We have presented a system that is able to perform multi-step tasks from demonstration, based on a non-parametric Bayesian model. We have demonstrated the system on a task that proved hard to program by hand, and shown that our model is robust to a large range of values of one of its meta-parameters. Compared to related work, our system is easier to deploy because it has less meta-parameters, while still providing good performances. We believe that this feature renders this work of interest for the programming-by-demonstration community and the robotic-system integrators.

References

[1] Sylvain Calinon, Florent Guenter, and Aude G. Billard. On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286–298, 2007.

[2] Sylvain Calinon, Florent D'Halluin, Eric L. Sauser, Darwin G. Caldwell, and Aude G. Billard. Learning and reproduction of gestures by imitation. *Robotics & Automation Magazine, IEEE*, 17(2):44–54, 2010.

[3] Manuel Mühlig, Michael Gienger, and Jochen J. Steil. Interactive imitation learning of object movement skills. *Autonomous Robots*, 32(2):97–114, 2012.

[4] George Konidaris asd Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[5] Jérome Maye, Rudolph Triebel, Luciano Spinello, and Roland Siegwart. Bayesian on-line learning of driving behaviors. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4341–4346. IEEE, 2011.

[6] Xianghai Wu and Jonathan Kofman. Human-inspired robot task learning from human teaching. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3334–3339. IEEE, 2008.

[7] Shuonan Dong and Brian Williams. Learning and recognition of hybrid manipulation motions in variable environments using probabilistic flow tubes. *International Journal of Social Robotics*, pages 1–12, 2012.

[8] Daniel H. Grollman and Odest Chadwicke Jenkins. Can we learn finite state machine robot controllers from interactive demonstration? In Olivier Sigaud and Jan Peters, editors, *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 407–430. Springer, 2010.

[9] Cédric Pradalier and Pierre Bessière. Perceptual navigation around a sensori-motor trajectory. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3831–3836. IEEE, 2004.

[10] Michael Bonani, Valentin Longchamp, Stéphane Magnenat, Philippe Rétornaz, Daniel Burnier, Gilles Roulet, Florian Vaussard, Hannes Bleuler, and Francesco Mondada. The marxbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *Proc. of the IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*, pages 4187–4193. IEEE, 2010.

[11] Frédéric Rochat, Patrick Schoeneich, Michael Bonani, Stéphane Magnenat, Francesco Mondada, Hannes Bleuler, and Christoph Hürzeler. Design of magnetic switchable device (MSD) and applications in climbing robot. In *Proc. of the 13th International Conference on Climbing and Walking Robots*, pages 375–382. World Scientific, 2010.

[12] Stéphane Magnenat, Roland Philippsen, and Francesco Mondada. Autonomous construction using scarce resources in unknown environments. *Autonomous Robots*, 33:467–485, 2012.

[13] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *Proc. of the IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*. IEEE, 2012.