Generalizing autonomously segmented complex trajectories based on learned task-specific environment metrics

Simon Hangl, Emre Ugur, and Justus Piater

University of Innsbruck, Email: firstname.lastname@uibk.ac.at

I. Introduction

We study how a robot can learn how to make best use of a number of demonstrated trajectories, which are effective for an arbitrary manipulation task in different environments, in order to accomplish the same task in a novel environment. In our previous work [3], we showed that the new trajectory can be computed as the weighted sum of the known trajectories (encoded as Dynamic Movement Primitives) where the weights are determined based on the similarities of the corresponding environments compared to the new one.
We discussed that pre-defining a metric to compare environment states is not realistic as the metric is not fixed and depends on task-related variables of the environment. For example the environment similarity depends on the size and geometry of the objects in an obstacle avoidance task whereas it depends on amount of liquid in a pouring task.
Thus, we proposed to learn this metric, representing it in Mahalanobis distance form, estimating it using the Euclidean distance between trajectories first, and finally refining it with policy reinforcement learning methods. When it comes to manipulation of real objects, just being able to generalize to new environments is not enough. Robots have to act in highly dynamic environments and have to be able to react on environment changes online as there are too many influencing components to be able to predict the appropriate reaction before executing the action. For example a tennis player may not predict external influences like wind, which yields a changed trajectory of the ball and therefore the number of mishits is increased on windy days, even for professional players. However, they are still able to adapt to the wind and other disturbances to some certain degree.
We showed how our control policy model can be used to easily adapt the manipulation according to changing environment. We demonstrated that the robot can effectively combine the set of trajectories to compute a new trajectory for new environments in tasks such as reaching and pouring using Kuka light weight robot arm.

In this paper, we extend our metric learning approach to complex tasks that involve multiple sub-tasks. We discuss that learning a single metric to compare two environment states for a complex task may not be sufficient as in different time-points of this task, different sub-task related environment state variables become important for the comparison. For example, in a complex task where obstacle avoidance is followed by pouring, the size of the obstacle is important first, and the amount of the liquid becomes important later. To deal with such situations, we propose a method that over-segments the trajectory, learns a metric for each segment using the original formulation, and combines the segments with similar metric values to obtain the final segmentation. Thus, our contribution is two-fold. First, based on learned metric similarity along the trajectory, the task can be segmented and subtasks can be automatically identified. Second, because more precise metrics are learned for each sub-task, this new formulation enables generalization of set of trajectories for new environments in complex tasks, whereas the non-segmented original formulation fails.

II. Task-specific Metric Learning

This section provides a summary of the task-specific metric learning approach that will be extended in this paper. For details of this approach, please see [3].

A. Overall model

Our method assumes a set of pairs of query space vectors and sample trajectories (modelled by some control policy like dynamic movement primitives [7]) solving a task under this state. Query space vector here corresponds to the task-related environment state variables. When the task has to be solved in a novel environment, a linear combination of the trajectories in the dataset is computed. The different instances have to be weighted differently, as not every sample has the same importance for the current situation (e.g. if a robot can pass right or left of a wall, simple averaging would lead to a crash).
Assuming that similar situations (i.e. similar query vectors) result in similar trajectories, the importance of a given sample can be determined by using a distance metric on the query space. As distance measure we use a Mahalanobis metric and learn the coefficients automatically by reusing ideas from *multidimensional scaling* [9] and reinforcement learning. The main advantage of formulating the new control policy as a linear combination is the ability to easily adapt the trajectory as soon as a change of the systems state is observed. This can be done by computing the new coefficients of the linear combination and smoothly transfer the values from the old to the new ones during execution.

B. Trajectory Blending

Let Z be a set of M sample trajectories associated with the query vector $\mathbf{q}_k$ (environment state in which this trajectory

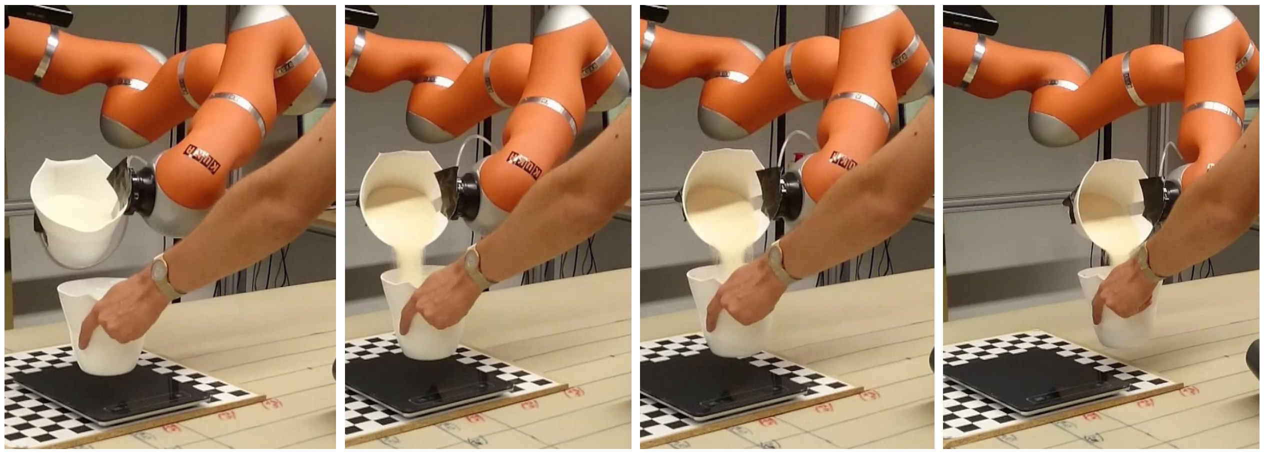

(a) Initial (b) Pouring (c) Pouring (d) Following

Fig. 1. Pouring experiment done with the original approach

solves the task) which are obtained through kinesthetic teaching. The query vector is assumed to capture enough information about the environment to decide, how the task should be solved (e.g. if the problem is a reaching task, the query vector would be a vector of Cartesian goal coordinates). Given such a set of trajectories with corresponding query vectors, we proposed a novel control policy model given by a linear combination of all trajectories. Under the assumption that these trajectories can approximate a manifold of all solutions for a given task, an arbitrary task-specific trajectory $\mathbf{y}_{\text{new}}$ solving the task for a novel situation can be approximated by

$$\mathbf{y}_{\text{new}}\left(\mathbf{q}_{\text{new}},t,\theta\right)=\frac{1}{s_{\text{reg}}}\sum_{i=1}^{M}s_i\mathbf{y}_i\left(\mathbf{q}_i,t,\theta_i\right) \tag{1}$$

where s_{reg} is a normalization term given by $s_{\text{reg}}=\sum_{i=1}^{M}s_i$ and $\mathbf{y}_i$ is a trajectory from the dataset. However, how to determine the weights s_i is a non-trivial problem in general, as it highly depends on the task. Given the assumption that similar situations can be handled by similar trajectories, s_i is assumed to be based on some similarity measure on the query space. The coefficients should have high values whenever a trajectory is more important than others for solving the current task. As it is assumed that the situation is fully described by the query/state vector, the coefficients can be defined by any strictly (with distance) decreasing function. This distance is measured by an arbitrary metric. As the query vectors $\mathbf{q}$ can have arbitrary dimensions and units, it is not straightforward to learn this metric for an arbitrary problem. One subclass of metrics is given by the *Mahalanobis distance*

$$m\left(\mathbf{q}_1,\mathbf{q}_2\right)=\sqrt{\left(\mathbf{q}_1-\mathbf{q}_2\right)^{\mathrm{T}}M\left(\mathbf{q}_1-\mathbf{q}_2\right)} \tag{2}$$

where $\mathbf{M}$ is a positive semi-definite matrix. The main idea is to learn the coefficients of the metric from the trajectories in the dataset, in order to determine the appropriate values for the coefficients s_i. It is important to note that with our proposed control policy formulation it is possible to *generalize* to novel situations by feeding the new query point (which reflects the current state of the environment) into the system. It is only required to re-estimate the coefficients s_i by using the previously learned similarity metric.

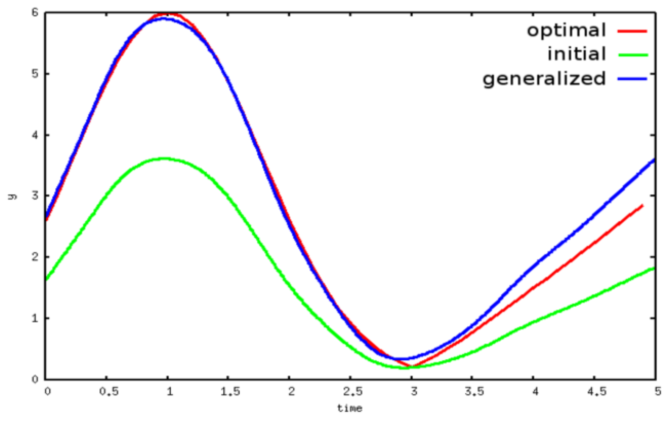
C. Metric Learning

As we use a metric to measure the importance of a sample trajectory for the currently observed environment state, the proposed control policy requires to learn this metric from the sample dataset. The metric is strongly dependent on the task and needs to be learned automatically.

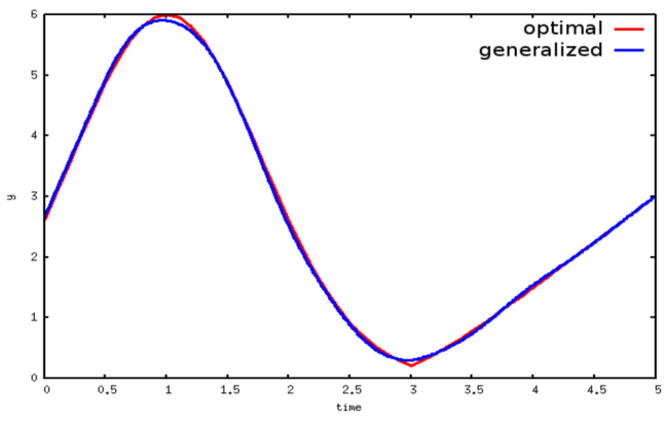
In order to perform a supervised metric learning approach to learn a Mahalanobis metric for $m(\mathbf{q}_i,\mathbf{q}_j)=m_{ij}$, where $\mathbf{q}_i,\mathbf{q}_j$ are query vectors and m_{ij} is the distance between these vectors, the values $\mathbf{q}_i$, $\mathbf{q}_j$, m_{ij} would be required. Unfortunately, the distances m_{ij} between the query vectors in the database are unknown. Therefore, a two step approach has been selected. A supervised metric learning algorithm using multidimensional scaling is used to compute an initial estimate by using trajectory similarity measures to compute m_{ij}.

Multidimensional scaling is a method to reconstruct a matrix $\mathbf{Q}$ of query vectors if the matrix $\mathbf{D}$ of pairwise squared distances between these vectors is known. Given that $\mathbf{Q}$ (query vectors of the trajectory database) and $\mathbf{D}$ (estimated from trajectory comparison) are already known, it is possible to estimate the coefficients of the Mahalanobis metric $m\left(\mathbf{q}_1,\mathbf{q}_2\right)$ instead of computing $\mathbf{Q}$. Please note that in this phase we only compute the values m_{ij} from trajectory similarity, while in fact these values belong to the distance of the query vectors.

This uses the assumption that similar trajectories are generated by control policies for similar query vectors. Even though this assumption might be too strong, it provides a good initial estimate of the metric m. However, this first step does not deliver a very precise estimate for the metric. Therefore, in a second step, the metric will be refined by using control policy reinforcement learning by



(a) Without segmentation: Method fails to model second part of the trajectory



(b) With segmentation: Method is able to reproduce ground truth trajectory

Fig. 2. Simulated trajectory for obstacle avoidance and subsequent reaching with the original (without segmentation) and the novel approach; green line: generalization with initial metric (no reinforcement learning), blue line: generalization after convergence of reinforcement learning, red line: ground truth

control policy reinforcement learning algorithms like PI^2 [8], PoWER [4] or policy gradient descent methods [6].

D. Trajectory Transitioning

Adaptive manipulation is especially important in case the manipulation of an object can result in the unexpected movement of this object due to the applied forces (e.g. screwing a nut, pushing an object) or in case that external forces can change the systems state unexpectedly (e.g. wind changes the trajectory of a tennis ball). Generalization to novel situations can be done by determining a new query vector $\mathbf{q}_{\text{next}}$ and re-estimating the coefficients using the learned metric as described above. Further, in order to make a smooth the transition between the current coefficients (s_i^{curr}) and next coefficients (s_i^{next}) we used the following update formula

$$s_i^{t_0+\Delta t} = s_i^{\text{next}} \left(1 - \mathrm{e}^{-\alpha_s \Delta t}\right) + s_i^{t_0} \mathrm{e}^{-\alpha_s \Delta t} \tag{3}$$

with the free parameter α_s.

III. Metric Learning with Autonomous Segmentation

In robotic manipulation, it often occurs that tasks are a concatenation of different action primitives. However, the sample data is provided as a single trajectory, where the borders between each primitive are not obvious to identify. We propose a new method to segment a trajectory at points, where the intrinsic structure of the problem changes. Therefore, we assume that action primitives differ in the geometry of the query space. This means that the metric changes over time if there is more than one primitive used to accomplish a task. We can use this information to identify action primitives that are meaningful for the corresponding manipulation task. This enables us to increase the precision of our method in more complex tasks in which the intrinsic problem changes over time.

To be able to identify the involved skills with this approach, the metric (and therefore coefficients s_i in Eq. (1)) have to be time dependent. In contrast to the original approach, we do not use a single metric but several metrics that are responsible for specific durations along the trajectory. These metrics are uniformly distributed over time. If two adjacent metrics differ strongly, they are considered to correspond to a different skill and therefore, a segmentation is done. Let $T := \{t_0, \ldots, t_N\}$ be a set of time points, then M_{t_k} is the used metric in the time range $R_k := [(t_k - t_{k-1})/2, (t_{k+1} - t_k)/2]$. Therefore, the distance between two query vectors $\mathbf{q}_i, \mathbf{q}_j$ at time $t \in R_k$ is given by

$$m_{\text{ext}}(\mathbf{q}_i, \mathbf{q}_j, t) = \sqrt{(\mathbf{q}_i - \mathbf{q}_j)^{\mathrm{T}} M_{t_k} (\mathbf{q}_i - \mathbf{q}_j)} \tag{4}$$

This extended metric $m_{\text{ext}}(\mathbf{q}_i, \mathbf{q}_j, t)$ replaces the metric $m(\mathbf{q}_i, \mathbf{q}_j)$ in the original approach. The learning of the metric is done similarly by initializing $\mathbf{M}_{t_i} = \mathbf{M}_{\text{init}}$ for all i and refining all metrics by reinforcing their coefficients with the approach described Section II. As soon as the new metric m_{ext} is learned, the distance between two neighbouring matrices is computed using *Frobenius norm*:

$$d_{i,(i+1)} = \|\mathbf{M}_{t_i} - \mathbf{M}_{t_{i+1}}\|_F$$

where Frobenius norm ($\|\|_F$) is defined as

$$\|A\|_F = \sqrt{\sum_k \sum_l A_{kl}^2} \tag{5}$$

If this distance is below a pre-set threshold for two neighbouring metrics, they are considered to belong to the same skill. If the distance is higher, a segmentation is done between the corresponding time points.

IV. Preliminary Results and Conclusion

In our previous work we evaluated the method without segmentation for a pouring task, where a certain amount of corn had to be poured from one cup into another. The position of the target cup was varied while manipulation. Snapshots of this experiment can be found in Fig. 1. To motivate our new approach we designed a simulated obstacle avoidance experiment which the original method was not able to solve. The robot should avoid an obstacle of varying height ($t \leq 3$) and do a reaching movement ($t \geq 3$) with different velocities after passing the obstacle (see Fig. 2). Therefore, the query vector is given by $\mathbf{q} = (h, v)$ with the obstacle height h and the velocity v. In Fig. 2(a) it can be seen that from $t = 3$ the original method is not able to perform the reaching movement properly, as a different metric would be required to perform it. On the other hand, with the new method, it is possible to properly reproduce the ground truth trajectory (see Fig. 2(b)). We learned the metric for 3 different time positions ($t \in \{1.0, 2.5, 4.0\}$). The algorithm delivered the metric results $\mathbf{M}_{1.0} = ((1, 0.9589), (0.9589, 0.9449))$, $\mathbf{M}_{2.5} = ((1, 0.9051), (0.9051, 0.8893))$, $\mathbf{M}_{4.0} = ((1, 0.7364), (0.7364, 4.1484))$. It can be seen that $\mathbf{M}_{1.0}$ and $\mathbf{M}_{2.5}$ are similar and therefore correspond to the same action primitive, while $\mathbf{M}_{4.0}$ differs from both. Therefore a segmentation has been performed. The result can be found in Fig. 2(b).

At the current stage of research, we uniformly over-segmented the trajectory, where the number of segmentation points has been defined manually. To provide a more meaningful over-segmentation other state of the art segmentation algorithms [1], [2], [5] could be used, where the extracted segments are then analysed by our method. The over-segmentation could also be determined by simple geometric properties (minima, maxima, ...) of the trajectory. It is important to note that the performance of the method strongly depends on the choice of the over-segmentation, as each additional segment introduces a number of further free parameters that have to be determined by reinforcement learning.

We presented an extension to our previous work on control policy metric learning that is able to segment a sample task to meaningful action primitives. This is done by introducing a time dependent metric, which replaces the metric in the original approach. Further, we showed that this approach significantly improves the generalization performance of our method.

Acknowledgement

The research leading to these results has received funding from the European Communitys Seventh Framework Programmes FP7/20072013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 610532, Squirrel and FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

References

[1] M. Buchin, A. Driemel, M. van Kreveld, and V. Sacristan. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science*, (3):33–63, 2011.

[2] M. Buchin, H. Kruckenberg, and A. Kölzsch. Segmenting trajectories based on movement states. In *Proc. 15th International Symposium on Spatial Data Handling (SDH)*. Springer, 2012. Accepted for publication.

[3] S. Hangl, E. Ugur, S. Szedmak, A. Ude, and J. Piater. Reactive, task-specific object manipulation by metric reinforcement learning. In *IEEE-RAS International Conference on Humanoid Robots*, 2014. submitted.

[4] J. Kober and J. Peters. Policy search for motor primitives in robotics. *Mach. Learn.*, 84(1-2):171–203, July 2011.

[5] S. Lee, I. Suh, S. Calinon, and R. Johansson. Autonomous framework for segmenting robot trajectories of manipulation task. *Autonomous Robots*, pages 1–35, 2014.

[6] J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE International Conference on Intelligent Robotics Systems (IROS 2006)*, 2006.

[7] S. Schaal. dynamic movement primitives - a framework for motor control in humans and humanoid robots. In *the international symposium on adaptive motion of animals and machines*, 2003.

[8] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.*, 11:3137–3181, Dec. 2010.

[9] F. W. Young. *Multidimensional Scaling: History, Theory, and Applications*. Lawrence, Erlbaum Associates, Publishers (Hillsdale, New Jersey; London), 1987.