A task-parameterized probabilistic model based on dynamical systems

Sylvain Calinon and Darwin G. Caldwell*

We use a *task-parameterized Gaussian mixture model* to encode the behavior of a virtual spring-damper system describing the movement of the robot, with variable stiffness and damping. The model adapts the centers and covariances of a GMM to the location and orientation of multiple objects and/or virtual landmarks, and allows transitions between different coordinate systems relevant to the task [1].

By setting $\boldsymbol{x}_n$, $\dot{\boldsymbol{x}}_n$ and $\ddot{\boldsymbol{x}}_n$ as position, velocity and acceleration of the end-effector at time t_n, it is assumed that the movement is driven by a virtual spring with stiffness $\boldsymbol{K}_n^{\mathcal{P}}$, damping $\boldsymbol{K}_n^{\mathcal{V}}$ and attractor point $\boldsymbol{y}_n$, similarly as in [2].

The *task parameters* are represented as P coordinate systems defined by $\{\boldsymbol{b}_{n,j}, \boldsymbol{A}_{n,j}\}_{j=1}^{P}$, representing respectively the origin of the observer and a transformation matrix (e.g., orientation). The movement is observed from these different viewpoints, forming a 3rd order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{D\times N\times P}$, composed of P trajectory samples $\boldsymbol{X}^{(j)} \in \mathbb{R}^{D\times N}$ of N time steps observed in P candidate frames.

For this benchmark, we have $D=5$ (time stamp, position of the end-effector and position of the attractor), and $P=2$ (candidate frames at the target and at the end-effector). We have $\boldsymbol{A}_{n,j} = \boldsymbol{I}\ \forall n, j$ (no modulation w.r.t. the orientation of the target/end-effector), $\boldsymbol{b}_{n,1} = [0, \boldsymbol{x}_n^{\mathcal{T}}, \boldsymbol{x}_n^{\mathcal{T}}]^\top$ and $\boldsymbol{b}_{n,2} = [0, \boldsymbol{x}_n, \boldsymbol{x}_n]^\top$.

The parameters of a model with K components (automatically determined by a Dirichlet process similarly as in [3]) are defined by $\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}\}_{i=1}^{K}$, where π_i are mixing coefficients that can optionally be omitted to speed up computation. $\boldsymbol{\mu}_i^{(j)}$ and $\boldsymbol{\Sigma}_i^{(j)}$ are the mode-j center and covariance matrix of the i-th Gaussian component. Learning of the parameters is achieved by maximizing the log-likelihood under the constraints that the data in the different frames are generated from the same source, resulting in an EM process to iteratively update the model parameters until convergence.

E-step:

$$\gamma_{n,i} = \frac{\pi_i \prod_{j=1}^{P} \mathcal{N}\big(\boldsymbol{X}_n^{(j)} |\ \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\big)}{\sum_{k=1}^{K} \pi_k \prod_{j=1}^{P} \mathcal{N}\big(\boldsymbol{X}_n^{(j)} |\ \boldsymbol{\mu}_k^{(j)}, \boldsymbol{\Sigma}_k^{(j)}\big)}.$$

M-step:

$$\pi_i = \frac{\sum_{n=1}^{N} \gamma_{n,i}}{N}\ , \quad \boldsymbol{\mu}_i^{(j)} = \frac{\sum_{n=1}^{N} \gamma_{n,i}\, \boldsymbol{X}_n^{(j)}}{\sum_{n=1}^{N} \gamma_{n,i}},$$

$$\boldsymbol{\Sigma}_i^{(j)} = \frac{\sum_{n=1}^{N} \gamma_{n,i}\, (\boldsymbol{X}_n^{(j)} - \boldsymbol{\mu}_i^{(j)})(\boldsymbol{X}_n^{(j)} - \boldsymbol{\mu}_i^{(j)})^\top}{\sum_{n=1}^{N} \gamma_{n,i}}. \qquad (1)$$

* Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 16163 Genova, Italy. sylvain.calinon@iit.it.

This work was supported by the AMARSi European project (FP7-ICT-248311) and by the STIFF-FLOP European project (FP7-ICT-287728).

The above algorithm is equivalent to the model presented in [1], but is computationally more efficient. The learned model can be used to reproduce movements in new situations (for new position and orientation of candidate frames). The system first retrieves at each time step n a GMM by a product of linearly transformed Gaussians

$$\mathcal{N}(\boldsymbol{\mu}_{n,i}, \boldsymbol{\Sigma}_{n,i}) \propto \prod_{j=1}^{P} \mathcal{N}\Big(\boldsymbol{A}_{n,j}\boldsymbol{\mu}_i^{(j)} + \boldsymbol{b}_{n,j},\ \boldsymbol{A}_{n,j}\boldsymbol{\Sigma}_i^{(j)}\boldsymbol{A}_{n,j}^\top\Big). \quad (2)$$

The system then relies on *Gaussian mixture regression* (GMR) [4], [5] to retrieve the attractor points and variations of the virtual spring. By defining which variables span for input and output parts (noted respectively by $\mathcal{I}$ and $\mathcal{O}$ superscripts), a block decomposition of the datapoints $\boldsymbol{\xi}_n$, vectors $\boldsymbol{\mu}_{n,i}$ and matrices $\boldsymbol{\Sigma}_{n,i}$ are written as

$$\boldsymbol{\xi}_n = \begin{bmatrix} \boldsymbol{\xi}_n^{\mathcal{I}} \\ \boldsymbol{\xi}_n^{\mathcal{O}} \end{bmatrix}\ , \quad \boldsymbol{\mu}_{n,i} = \begin{bmatrix} \boldsymbol{\mu}_{n,i}^{\mathcal{I}} \\ \boldsymbol{\mu}_{n,i}^{\mathcal{O}} \end{bmatrix}\ , \quad \boldsymbol{\Sigma}_{n,i} = \begin{bmatrix} \boldsymbol{\Sigma}_{n,i}^{\mathcal{I}} & \boldsymbol{\Sigma}_{n,i}^{\mathcal{I}\mathcal{O}} \\ \boldsymbol{\Sigma}_{n,i}^{\mathcal{O}\mathcal{I}} & \boldsymbol{\Sigma}_{n,i}^{\mathcal{O}} \end{bmatrix}.$$

At each reproduction step, the conditional probability $\mathcal{P}(\boldsymbol{\xi}_n^{\mathcal{O}} | \boldsymbol{\xi}_n^{\mathcal{I}})$ is estimated with GMR as an output distribution $\mathcal{N}(\hat{\boldsymbol{\mu}}_n^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}_n^{\mathcal{O}})$, that is also Gaussian. Learning the model depends linearly on the number of datapoints. Prediction and reproduction is independent on the number of datapoints in the training set. During reproduction, retrieving statistical information about the outputs corresponds to a convex sum of linear systems, making the algorithm computationally light.

The variability and correlation information are then exploited in a *linear quadratic regulator* (LQR) to retrieve smooth acceleration commands adapting the tracking of the target to the accuracy requirements, varying during the movement. The details of the overall approach are presented in Alg. 1.

REFERENCES

[1] S. Calinon, T. Alizadeh, and D. G. Caldwell, "Improving extrapolation capability of task-parameterized movement models," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, November 2013.

[2] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, Osaka, Japan, 2012, pp. 323–329.

[3] D. Bruno, S. Calinon, and D. G. Caldwell, "Bayesian nonparametric multi-optima policy search in reinforcement learning," in *Proc. AAAI Conference on Artificial Intelligence*, Bellevue, Washington, USA, 2013, pp. 1374–1380.

[4] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an EM approach," in *Advances in Neural Information Processing Systems*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan Kaufmann Publishers, Inc., 1994, pp. 120–127.

[5] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 37, no. 2, pp. 286–298, 2007.

1. Learning

Input: Time-indexed movement demonstrations and (moving) target $\{t_n, \boldsymbol{x}_n, \dot{\boldsymbol{x}}_n, \ddot{\boldsymbol{x}}_n, \boldsymbol{x}_n^{\mathcal{T}}\}_{n=1}^{N}$.

Output: Task-parameterized mixture model parameters $\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}\}_{i=1}^{K}$.

1.1 Estimate K *(nb comp.)* with Dirichlet process, set $\boldsymbol{K}_n^{\mathcal{P}} = \kappa^{\mathcal{P}} \boldsymbol{I}$ *(initial stiffness)*, $\boldsymbol{K}_n^{\mathcal{V}} = 2\sqrt{\kappa^{\mathcal{P}}} \boldsymbol{I}$ *(initial damping)*.

1.2 Compute attractor point $\boldsymbol{y}_n$ for each n with $\boldsymbol{y}_n = (\boldsymbol{K}_n^{\mathcal{P}})^{-1} \ddot{\boldsymbol{x}}_n + (\boldsymbol{K}_n^{\mathcal{P}})^{-1} \boldsymbol{K}_n^{\mathcal{V}} \dot{\boldsymbol{x}}_n + \boldsymbol{x}_n$.

1.3 Fit a model to $\{\boldsymbol{\xi}_n = [t_n, \boldsymbol{x}_n, \boldsymbol{y}_n]^\top\}_{n=1}^{N}$ with Eq. (1), modeling $\boldsymbol{\xi}_n^{\mathcal{I}} = [t_n, \boldsymbol{x}_n]^\top$ and $\boldsymbol{\xi}_n^{\mathcal{O}} = \boldsymbol{y}_n$ with Eq. (2) as $\boldsymbol{\xi}_n^{\mathcal{I}}, \boldsymbol{\xi}_n^{\mathcal{O}} \sim \sum_{i=1}^{K} \pi_i \, \mathcal{N}(\boldsymbol{\mu}_{n,i}, \boldsymbol{\Sigma}_{n,i})$.

2. Reproduction

Input: Model parameters $\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}\}_{i=1}^{K}$, initial point $\hat{\boldsymbol{x}}_0$ and (moving) target $\boldsymbol{x}_n^{\mathcal{T}}$.

Output: Retrieved movement $\{\hat{\boldsymbol{x}}_n, \hat{\dot{\boldsymbol{x}}}_n, \hat{\ddot{\boldsymbol{x}}}_n\}_{n=1}^{T}$.

for $n \leftarrow T$ **to** 1 *(backward recursion)*

2.1 Use GMR to predict attractor distribution $\mathcal{N}(\hat{\boldsymbol{y}}_n, \hat{\boldsymbol{\Sigma}}_n^{\boldsymbol{y}})$ with $\hat{\boldsymbol{y}}_n = \sum_{i=1}^{K} h_{n,i}(\boldsymbol{\xi}_n^{\mathcal{I}}) \left[\boldsymbol{\mu}_{n,i}^{\mathcal{O}} + \boldsymbol{\Sigma}_{n,i}^{\mathcal{OI}} {\boldsymbol{\Sigma}_{n,i}^{\mathcal{I}}}^{-1} (\boldsymbol{\xi}_n^{\mathcal{I}} - \boldsymbol{\mu}_{n,i}^{\mathcal{I}})\right]$ and $\hat{\boldsymbol{\Sigma}}_n^{\boldsymbol{y}} = \sum_{i=1}^{K} h_{n,i}^2(\boldsymbol{\xi}_n^{\mathcal{I}}) \left[\boldsymbol{\Sigma}_{n,i}^{\mathcal{O}} - \boldsymbol{\Sigma}_{n,i}^{\mathcal{OI}} {\boldsymbol{\Sigma}_{n,i}^{\mathcal{I}}}^{-1} \boldsymbol{\Sigma}_{n,i}^{\mathcal{IO}}\right]$, where $\boldsymbol{\mu}_{n,i}$ and $\boldsymbol{\Sigma}_{n,i}$ are computed in Eq. (2).

2.2 Use LQR to estimate $\boldsymbol{K}_n^{\mathcal{P}}$ and $\boldsymbol{K}_n^{\mathcal{V}}$ in the system $\boldsymbol{u}_n = \hat{\ddot{\boldsymbol{x}}}_n = \boldsymbol{K}_n^{\mathcal{P}}(\hat{\boldsymbol{y}}_n - \boldsymbol{x}_n) - \boldsymbol{K}_n^{\mathcal{V}} \dot{\boldsymbol{x}}_n$, with backward integration of the Riccati equation corresponding to the minimization of the cost function $\sum_{n=1}^{T} (\boldsymbol{x}_n - \hat{\boldsymbol{y}}_n)^\top \boldsymbol{Q}_n (\boldsymbol{x}_n - \hat{\boldsymbol{y}}_n) \; + \; \boldsymbol{u}_n^\top \boldsymbol{R} \, \boldsymbol{u}_n$, with varying weighting term $\boldsymbol{Q}_n = (\hat{\boldsymbol{\Sigma}}_n^{\boldsymbol{y}})^{-1}$ computed with step **(2.1)**.

end

for $n \leftarrow 1$ **to** T *(forward prediction)*

2.3 Update task parameters $\boldsymbol{b}_{n,j}$ to possible changes of target position, and recompute $\boldsymbol{\mu}_{n,i}$ with Eq. (2).

2.4 Compute acceleration command $\hat{\ddot{\boldsymbol{x}}}_n = \boldsymbol{K}_n^{\mathcal{P}}(\hat{\boldsymbol{y}}_n - \boldsymbol{x}_n) - \boldsymbol{K}_n^{\mathcal{V}} \dot{\boldsymbol{x}}_n$ by using steps **(2.1)** and **(2.2)**.

2.5 Update velocity $\hat{\dot{\boldsymbol{x}}}_n$ and position $\hat{\boldsymbol{x}}_n$ of the end-effector.

end

Algorithm 1: Proposed approach.