Reward-weighted GMM and its application to action-selection in robotized shoe dressing

Adrià Colomé, Sergi Foix, Guillem Alenyà and Carme Torras

Institut de Robòtica i Informàtica Industrial CSIC-UPC
Llorens i Artigas 4-6, 08028, Barcelona, Spain.
[acolome, sfoix, galenya, torras {@iri.upc.edu}]

Abstract. In the context of assistive robotics, robots need to make multiple decisions. We explore the problem where a robot has multiple choices to perform a task and must select the action that maximizes success probability among a repertoire of pre-trained actions. We investigate the case in which sensory data is only available before making the decision, but not while the action is being performed. In this paper we propose to use a Gaussian Mixture Model (GMM) as decision-making system. Our adaptation permits the initialization of the model using only one sample per component. We also propose an algorithm to use the result of each execution to update the model, thus adapting the robot behavior to the user and evaluating the effectiveness of each pre-trained action. The proposed algorithm is applied to a robotic shoe-dressing task. Simulated and real experiments show the validity of our approach.

1 Introduction

Assistive robots are becoming a reality. Apart from the safety requirements, one of the main issues is related to the high variability of human environments. In this context, perceptions, and by extension the measurement of the current state, are partial and uncertain. Additionally, the effects of the actions of the robot are also uncertain, as we consider that actions may not only succeed or fail, but also have intermediate results.

In the framework of the I-DRESS project, we are investigating how robots could be useful assistants in dressing tasks. These tasks require interaction with the user, adaptation to the preferences of the user, and guaranteeing safety.

In assistive robotics, the ability to take adequate decisions is crucial. During the completion of a task, a robot has to decide, multiple times, among different actions. In this paper we put the focus on choosing among actions that solve the same goal but use different strategies.

Let us use an example to provide further intuition. The task of shoe dressing is quite complex: it involves several actions (choose a shoe, grasp, approach, insert); different types of shoe require different insertion strategies; and people have different mobility restrictions. It is very difficult to program a robot to cope with all this variability. We envisage that this can be done by providing

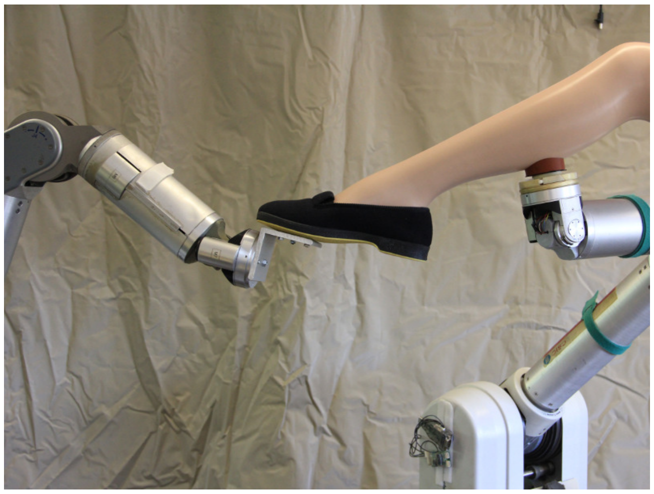

Fig. 1: Overall setup.

the robot with a repertoire of strategies and the ability of choosing one among them.

Here we concentrate on the insertion action as it provides a perfect example (see Fig. 1). Imagine that the robot has a repertoire of *insert* actions, learned in the factory environment, which are suited to several foot orientations of reference. In a real situation, the robot has to decide which *insert* action to execute based on the current orientation of the user foot. Naturally, the current orientation will not perfectly correspond to a reference orientation, so an estimation of the most convenient one is required.

When the relevant sensory data is available during the execution of the trajectory at a high rate, the usual strategy is to define an error function and reshape the trajectory to minimize this error function. In a recent work, Pignat and Calinon [7] propose a method that allows to demonstrate few example trajectories that include sensory data. Their system is able not only to reproduce the expected behavior, but also to learn the relevant relationships between sensory data and robot motion. Note that visual information is most often used, but in assistive robotics force plays an important role. In this paper we explore the case when this information is not available. We assume that the only information available are some snapshots at the beginning and the end of the actions, so an error function cannot be defined.

Action selection (also known as the problem of what to do next) is a broad field in artificial intelligence. The available selection methods can be divided in two families: global and local. Global methods use the information about the long-term goal, the set of actions that the robot can execute, and their effects and costs, to obtain a plan that optimizes the cost of the entire set of actions. Kaelbling and Lozano-Perez [5] proposed a hierarchical planning algorithm to limit the length of the obtained plans. Martínez *et al.* [6] proposed to use short-horizon planning in tasks where the uncertainty of the actions makes the prediction of the state too ambiguous after few actions.

Contrarily, local methods use only the information at a given point to choose the action that maximizes the immediate reward. Local action selection has been used traditionally in active vision to decide the placement of the camera (see [2] for a review). Different measurements can be used as decision criterion. For example, Foix *et al.* [3] propose to use a variation of the information gain to combine gains from two different sources. All these methods are based on estimating the effects of executing given actions. However, here we deal with a slightly different problem, as the only two possible outcomes of the actions are success/not_success, and the problem is to select the appropriate action that maximizes the success probability.

In this paper we propose a decision-making system based on an adaptation of a Gaussian Mixture Model (GMM) that permits initialization by using only one sample per component. We also show how the model can be updated after each new execution by taking advantage of the result of the preceding trial.

2 Revisiting Gaussian Mixture Models

A Gaussian Mixture Model (GMM) distribution over a random variable $\mathbf{x}$ can be written as a weighted superposition of K Gaussians with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$, weighted by their mixing proportions π_k [1], for $k = 1..K$:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{1}$$

where it is common to consider π_k as the probability of a K-dimensional random variable $\mathbf{z}$, with $z_k \in \{0,1\}$ and $p(z_k = 1) = p(z_k = 1, z_{\neq k} = 0) = \pi_k$ [1]. Note that $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$. Therefore, $p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and, marginalizing $\mathbf{x}$ wrt. $\mathbf{z}$ we obtain Eq. (1) again:

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2}$$

The GMM model in Eq.(1) can be obtained with an Expectation-Maximization (EM) algorithm [1], for which we need to compute a term $\gamma(z_{ik}) \triangleq p(z_k = 1|\mathbf{x}_i)$ by using the Bayes' rule:

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum\limits_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}. \tag{3}$$

$\gamma(z_{ik})$ is called the *responsibility* of the component k associated with a sample $\mathbf{x}_i$, and we will use it later for action selection.

Using the aforementioned EM algorithm with a set of N data samples $\mathbf{X}$, the log-likelihood $\ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be maximized in an iterative 2-step optimization:

– **E-step**: Evaluate the responsibilities using current parameters with Eq.(3)
– **M-step**: Re-calculate the parameters with (see [1]):

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k}\sum_{i=1}^{N}\gamma(z_{ik})\mathbf{x}_i \tag{4}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k}\sum_{i=1}^{N}\gamma(z_{ik})(\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})^T \tag{5}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{6}$$

where $N = \sum_k N_k$ and

$$N_k = \sum_{i=1}^{N}\gamma(z_{ik}) \tag{7}$$

– **Evaluate** the log-likelihood:

$$\ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N}\ln\left[\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \tag{8}$$

and check for convergence.

3 Using a GMM for action selection

In this section, we adapt a GMM in order to be used for action selection. For this purpose, we match the number of Gaussian components K to the number of actions available, and use the samples $\mathbf{x} \in \mathbb{R}^d$ as contextual features that will take the role of variables deciding which action to execute, d being the dimension of the feature space.

We modify the GMM defined in Section 2, so that it can be initialized with fewer samples (even as few as one sample per component). Such small number of samples results in the need to adapt the values N_k, as the responsibilities or number of samples to generate the distribution might not be available. Moreover, we add weights to the GMM responsibilities in order to update the model after each sample, according to the results of the experimentation. Such procedure helps to improve the decision-making algorithm.

3.1 Initializing a GMM with few samples

In order to initialize a GMM with few samples, we assume we have K posible actions, with m_k samples for each of them, and we know the action associated with each sample in the training dataset. If $m_k \geq 4d$, it is then considered that there is probably enough data to fit the covariance matrix of mixing the component k [4]. In that case, the EM algorithm presented in Section 2 generates a properly initialized model. Otherwise, We can initialize the means of the GMM by using the average of the samples associated to each action $k = 1..K$:

$$\boldsymbol{\mu}_k = \frac{1}{m_k} \sum_{i \in K} \mathbf{x}_i. \tag{9}$$

The covariances for each component are then initialized by assuming a 1.5 *standard deviations intersection*. That is, $\boldsymbol{\Sigma}_k = \lambda_k^2 \mathbf{I}$, where

$$\lambda_k = \min_{j=1..K, j \neq k} \frac{\|\boldsymbol{\mu}_j - \boldsymbol{\mu}_k\|}{1.5}. \tag{10}$$

Last, we initialize $\pi_k = 1/K$, $\forall k = 1..K$.

The rationale behind Eq.(10) is that the resulting Gaussian components are intersecting, but with an overlap small enough to be discriminative. We build hyperspheres in a d-dimensional space, centered at the mean points $\boldsymbol{\mu}_k$ (Eq (9)) and radii so that their contacts are at most tangent. Then, we use these radii and set them to be 1.5 standard deviations on an isotropic covariance matrix, as defined in Eq. (10). We found that the 1.5 ratio offered a good trade-off between Gaussians not being too small and having a relevant probability in their region of overlap. Figure 2 shows an example of the proposed initialization of the GMM with only 3 samples per component in a GMM with $K = 5$ components, comparing different values of the number of standard deviations-equivalence used in Eq.(10).

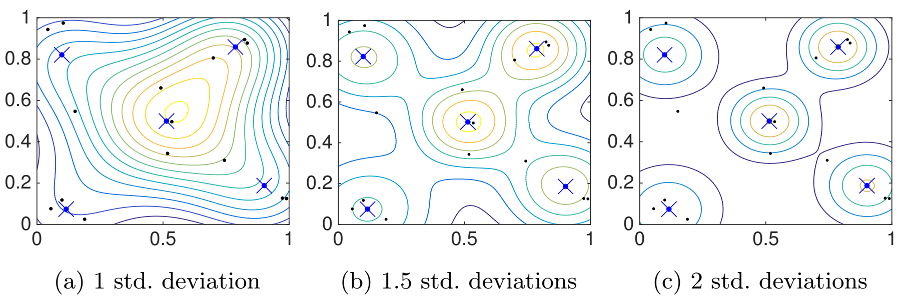

Fig. 2: Initialization of a GMM with few samples. The means of the components (blue crosses) are obtained by averaging the samples (black dots). Then, the covariance matrix is obtained with Eq.(10). As we can see, 1 standard deviation generates a GMM with too much overlap between components, while 2 standard deviations have a too reduced overlap. Therefore, we used 1.5 standard deviations equivalence.

3.2 Reward-Weighted Responsibility GMM (RWR-GMM)

In order to apply GMMs to action selection, we now present a variant of the GMM presented in Section 2, where the responsibilities $\gamma(z_{sk})$ play a crucial role. Given a sample $\mathbf{x}_s$, the action to execute will be selected depending on such responsibility values, as $\gamma(z_{sk}) = p(z_k = 1|\mathbf{x}_s)$. However, the GMM needs to be adapted after each execution, depending on the success of the decision taken by our selector. For this reason, if the success is denoted by the random variable $r \in [0, 1]$, such that $r_{sk} = 1$ if action k was successful when applied in situation $\mathbf{x}_s$, then we define the *reward-weighted responsibility* of an action as

$$d_{sk}\gamma(z_{sk}) = p(r_{sk} = 1|z_k = 1, \mathbf{x}_s)p(z_k = 1|\mathbf{x}_s), \tag{11}$$

which is equivalent to the probability of success when applying action k in a situation described by $\mathbf{x}_s$.

Given such weighted responsibilities, we can substitute $d_{sk}\gamma(z_{sk})$ for $\gamma(z_{sk})$ in Eqs.(4)-(7) and obtain a Reward-Weighted Responsibility GMM (RWR-GMM):

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} d_{sk}\gamma(z_{sk})\mathbf{x}_i \tag{12}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} d_{sk}\gamma(z_{sk})(\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})^T \tag{13}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{14}$$

where $N = \sum_k N_k$ and

$$N_k = \sum_{s=1}^{N} d_{sk}\gamma(z_{sk}) \tag{15}$$

This permits assigning importance to samples, but it is specially used to update the model in a one-step manner, as will be described in Section 3.3.

3.3 Updating an RWR-GMM

Another issue we can encounter is how to update a given RWR-GMM $(\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ when a new sample is added and we do not know how many samples were used to build it, or if it was artificially generated. This results in unknown values for the responsibilities $\gamma(z_{sk})$ and N_k. However, given Eq. (7), we can assume that the GMM was generated with a certain number of samples N and so, isolating N_k from Eq.(7), $N_k = N\pi_k$. The choice of N will depend on the relative importance we want to give to a newly-generated sample and acts as a forgetting factor. For example, setting $N = 100$ would be equivalent to treat the GMM as if it was generated with such amount of samples, and the effect of an additional sample would be relative to that amount.

If, given a model $(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, we add a sample $\mathbf{x}_n$, equations (12)-(15) can be reformulated by adding a point and substituting the old values of $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$:

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{N_k}{N_k + d_{sk}\gamma(z_{sk})} \left(\boldsymbol{\mu}_k + \frac{d_{sk}\gamma(z_{sk})\mathbf{x}_n}{N_k} \right) \tag{16}$$

Algorithm 1 RWR-GMM using all actions

Input: GMM with $\{\pi_k, \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k\}$ and a preset value of N, so that $N_k = N\pi_k$
Output: Updated GMM parameters $\{\pi_k, \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k\}$

```
1: for  $s = 1 : N_{\text{samples}}$ do
2:     For the given sample $\mathbf{x}_s$, execute all the actions $k = 1..K$ and check whether
       the execution was successful.
3:     Update the RWR-GMM with $d_{sk} = 1$, $\forall k$ that were successful, and $d_{sk^\star} = 0$
       otherwise.
4: end for
```

Algorithm 2 RWR-GMM checking only one action

Input: GMM with $\{\pi_k, \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k\}$ and a preset value of N, so that $N_k = N\pi_k$
Output: Updated GMM parameters $\{\pi_k, \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k\}$

```
1: for  $s = 1 : N_{\text{samples}}$ do
2:     Compute the responsibilities $\gamma(z_{sk})$, $\forall k = 1..K$, for the given sample $\mathbf{x}_s$ with
       Eq. (3). Note that $\sum_k \gamma(z_{sk}) = 1$.
3:     Randomly select an action $k^\star$ using the probabilities obtained with responsibil-
       ities
4:     Execute the action $k^\star$ and check if it was successful.
5:     If action $k^\star$ was successful, we update the RWR-GMM with $d_{sk} = 0$, $\forall k \neq k^\star$,
       $d_{sk^\star} = 1$. Otherwise, $d_{sk} = 1$, $\forall k \neq k^\star$ and $d_{sk^\star} = 0$.
6: end for
```

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{N_k}{N_k + d_{sk}\gamma(z_{sk})} \left(\boldsymbol{\Sigma}_k + \frac{d_{sk}\gamma(z_{sk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T}{N_k} \right) \tag{17}$$

$$N_k^{\text{new}} = N_k + d_{sk}\gamma(z_{sk}) \tag{18}$$

In practical applications, we will have a number K of actions to select from and, given an initialized GMM as in Section 3.1, a new sample $\mathbf{x}_s$ consisting of state feature variables that can be observed/measured. We propose two alternatives for improving the method: in the first one, we assume we can test all the possible actions for each point $\mathbf{x}_s$, and we will set $d_{sk} = 1$ for the successful actions and 0 otherwise (see Alg. 1). Alternatively, in case the assumption is not satisfied, we will evaluate the responsibilities of each action for each new sample $\mathbf{x}_s$, and randomly choose an action $k^\star$ with a probability equal to its responsibility. Then, this action will be executed and tested for success. If so, its weight will be set to $d_{sk^\star} = 1$ and the other actions to 0. Otherwise, $d_{sk^\star} = 0$ and, for the other actions, it will be set to 1, as shown in Alg. 2.

After setting the reward weights, we can update the model with Eqs. (16)-(18).

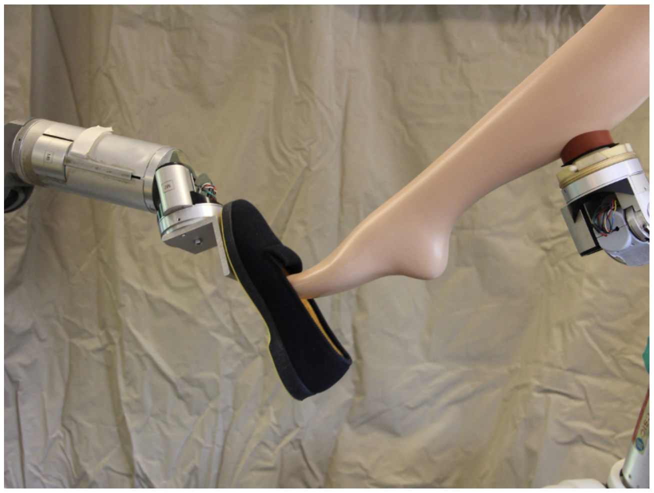

Fig. 3: Initial pose for each of the shoe insertion trajectories. Notice that the tip of the foot is already into the shoe.

4 Experimentation

Figure 1 shows the overall setup used in our experiments. In order to ensure a good initialization pose for both shoe and foot, experiments were carried out using two arm manipulators. As it can be seen, we used a mannequin leg instead of a real human leg. Thanks to that, we managed to avoid two complicated issues: the use of an external vision system for a precise foot pose estimation and the difficulty for a human to repeatedly maintain a constant foot pose while the robot is putting the shoe. Figure 3 shows a detailed view of the initial pose for each of the demonstration trajectories. It is important to notice that both the tip of the foot and the shoe poses are always in the same initial position for all the experiments no matter the orientation. This way, we ensure that a wrong action cannot be due to failed shoe insertion.

Obviously, the action required to insert a shoe depends on the position of the foot but also on the type of shoe. Shoes like flip flops and some sandals require easy motions as do not have a back strap. Contrarily, some orthopedic shoes are challenging because the topline on top of the counter can bend over. Observe that the shoe used in the experiments (Fig. 1) belongs to the last group, and that means that *insert* actions should tend to fail more easily.

Experiments consisted of the following steps: first, we taught the robot, through kinesthetic teaching, how to perform a good shoe insertion for each of the reference demonstration foot poses; and second, we equally divided the configuration space by 0.1 *rad.* and tested exhaustively each intermediate foot pose, $\mathbf{x}_s = \begin{bmatrix} \text{pitch}_s \\ \text{yaw}_s \end{bmatrix}$, with each of the taught shoe insertion actions. Foot poses were chosen as realistic as possible and by taking into account a range foot orientation comfortable for humans when performing that task, ± 0.5 radian range for both pitch and yaw angles. Given the bi-dimensionality of the parametrization and the short ranges, we trained the system with 5 different demonstration poses. One central and the other 4 located at the limit of each range. Using the robot that was holding the mannequin leg as a reference frame, these are the selected centers for each GMM:

$$\begin{aligned} \boldsymbol{\mu}_1 &= [0; -1.5], \\ \boldsymbol{\mu}_2 &= [-0.5; -1.5], \\ \boldsymbol{\mu}_3 &= [0.5; -1.5], \\ \boldsymbol{\mu}_4 &= [0; -2.0], \\ \boldsymbol{\mu}_5 &= [0; -1.0]. \end{aligned} \tag{19}$$

After using the initialization method described in Sec.3.1, we obtain the initialization map shown in Figure 4. Observe the homogeneous distribution of the Gaussians over the central pose point and the 4 limit pose points. Figure 5 shows the success of actions after testing for each action at each intermediate point $\mathbf{x}_s$ (pitch $= -0.5 : 0.1 : 0.5$ and yaw $= -2 : 0.1 : -1$), in a total of 605 executions. Notice how actions taught at different foot pitch angles overlap on some intermediate poses, showing a good compatibility; the incompatibility between actions taught at different foot yaw angles can also be seen.

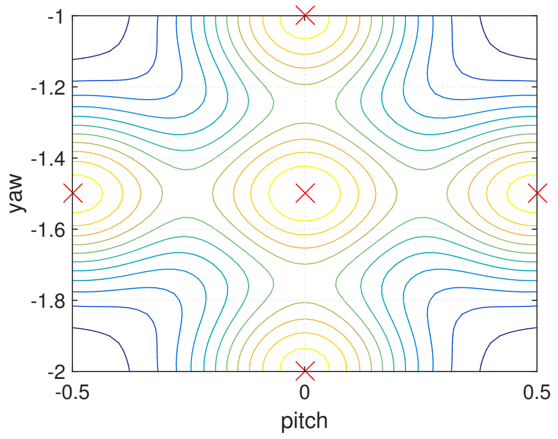

Fig. 4: Initialization of the GMM with only one sample per node.

Whether a shoe insertion action was successfully achieved or not was controlled by visual check. Please see Figures 6a, b for a good and a bad example, respectively.

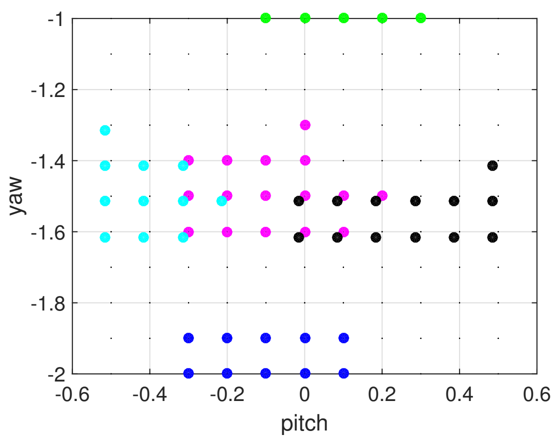

Fig. 5: Successful executions for each of the 5 actions.

4.1 Using Algorithm 1

Among the 121 tested points, we only considered those 46 which were successful. Among these 46, we used 21 points for training, and the remaining 25 for validation. Using the initialization in Figure 4, with an initial value of $N = 20$, and training the model following Algorithm 1, we compared the successful actions for the validation set with the responsibilities obtained from the RWR-GMM updates, see Fig. 7. In order to evaluate the accuracy of the system, we fitted the model with 10 randomly chosen sample subsets and checked its responsibility for an action. After evaluating all of them and averaging the result, we obtained 98.4% accuracy in assigning the highest responsibility to an action that was successful. The remaining 1.6% were failures corresponding to points where the successful action was the second with the highest responsibility, with a value of more than 0.3, therefore corresponding to overlapped regions.

4.2 Using Algorithm 2

Similarly to the previous subsection, we took 21 of the points with at least one successful action and trained the system using the procedure described in Alg.2, see Fig. 8. In this case, we obtained 98% success in assigning the highest probability to a successful action, the unsuccessful ones also being in positions where a successful action had a responsibility higher than 0.3.

5 Conclusions

In this article we have shown how a GMM can be adapted to be used as an action decision-making system. Such adaptation permits shortening the model initialization process, by reducing the number of required samples thanks to

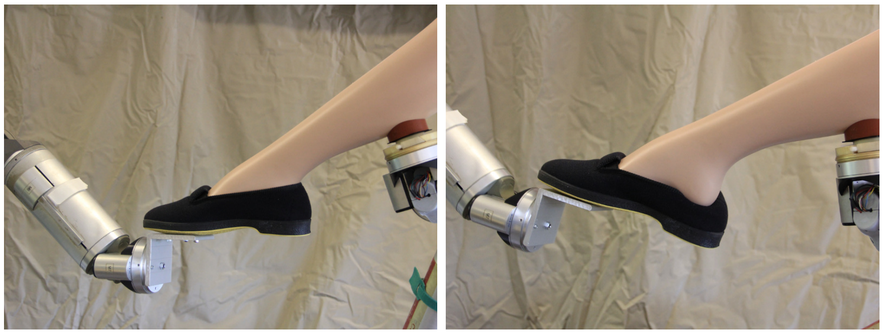
(a) Success. (b) Failure.

Fig. 6: Visual check of action fulfillment. a) shows a perfectly successful action. b) shows a non-satisfactory case.

some minor assumptions, and to improve the update of the model at runtime, by adding rewards as weights into the GMM responsibilities. Experiments have demonstrated the validity of the approach by achieving 98% accuracy in a very challenging real scenario, shoe dressing. Two algorithms have been presented, one that considers observability of all possible actions for testing at each step and another that evaluates only one at a time. We have shown how the latter, although slightly less precise, is highly efficient and requires 5 times less executions.

As future work we would like to extend experimentation to other user-adjustable robotic assistive tasks such as jacket or gown dressing, food feeding, or prosthetic grasping. To carry out such complex tasks, the proposed system could be enhanced by making robot skills feature-dependent, so as to better adapt robot motion to the observed situation.

References

1. C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2007.
2. S. Chen, Y. Li, W. Wang, and J. Zhang. *Active sensor planning for multiview vision tasks*, volume 1. Springer-Verlag, Berlin Heidelberg, 2008.
3. S. Foix, G. Alenyà, and C. Torras. 3D sensor planning framework for leaf probing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6501–6506, 2015.
4. N. Hansen. The CMA evolution strategy: A tutorial. *Computer Research Repository*, 2016.
5. L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1470–1477. IEEE, 2011.
6. D. Martínez, G. Alenyà, and C. Torras. Planning robot manipulation to clean planar surfaces. *Engineering Applications of Artificial Intelligence*, 39:23–32, 2015.
7. E. Pignat and S. Calinon. Learning adaptive dressing assistance from human demonstration. *Robotics and Autonomous Systems*, 93:61–75, July 2017.

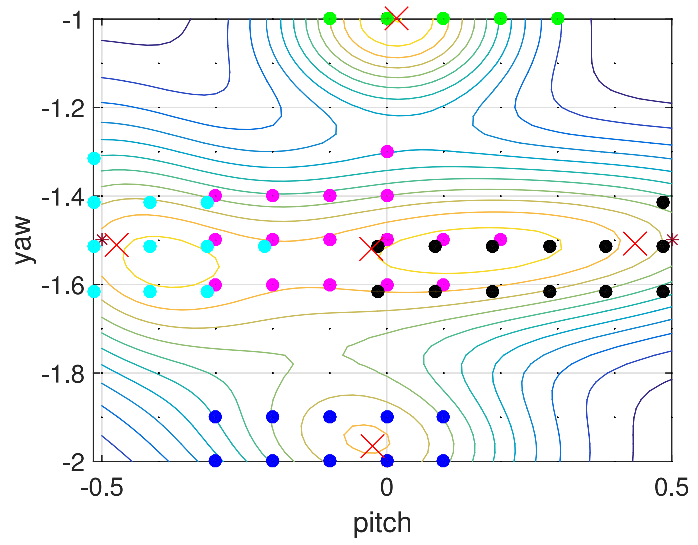

Fig. 7: Results with sampling as in Alg.1.

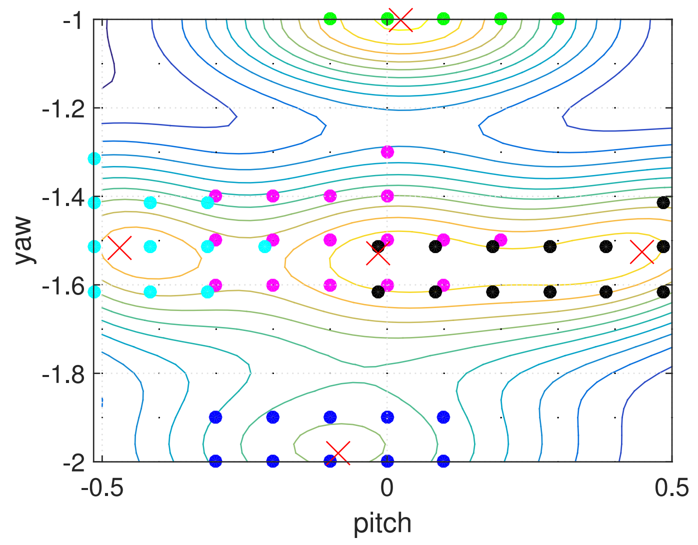

Fig. 8: Results with sampling as in Alg.2.