Joining high-level symbolic planning with low-level motion primitives in adaptive HRI: application to dressing assistance

Gerard Canal*[1], Emmanuel Pignat*[2], Guillem Alenyà[1], Sylvain Calinon[2] and Carme Torras[1]

***Abstract*— For a safe and successful daily living assistance, far from the highly controlled environment of a factory, robots should be able to adapt to ever-changing situations. Programming such a robot is a tedious process that requires expert knowledge. An alternative is to rely on a high-level planner, but the generic symbolic representations used are not well suited to particular robot executions. Contrarily, motion primitives encode robot motions in a way that can be easily adapted to different situations. This paper presents a combined framework that exploits the advantages of both approaches. The number of required symbolic states is reduced, as motion primitives provide "smart actions" that take the current state and cope online with variations. Symbolic actions can include interactions (e.g., ask and inform) that are difficult to demonstrate. We show that the proposed framework can adapt to the user preferences (in terms of robot speed and robot verbosity), can readjust the trajectories based on the user movements, and can handle unforeseen situations. Experiments are performed in a shoe-dressing scenario. This scenario is particularly interesting because it involves a sufficient number of actions, and the human-robot interaction requires the handling of user preferences and unexpected reactions.**

I. INTRODUCTION

Healthcare assistance is often related to close-contact interactions. Caregivers frequently help users to get dressed, eat or clean when they cannot do these tasks by themselves. These Activities of Daily Living (ADLs) are basic needs of every human being, and robots may help users to perform them more autonomously. However, these ADLs are complex tasks that often require close-contact physical interactions that can be dangerous when performed by the robot.

Physical Human-Robot Interaction (pHRI) is a special case of HRI in which safety becomes a central issue due to the possibility of potentially causing harm to a human user. Therefore, physically assistive robots need to be equipped with two basic skills: *compliant control* to ensure safe motion, and *planning* taking the user into consideration in order to foresee possible problems and deviations in the execution of the task.

This work has been supported by the ERA-Net CHIST-ERA project I-DRESS (Spain's Ministry of Economy and Competitiveness PCIN-2015-147, Swiss National Science Foundation 20CH21-16085). It is also supported by the Spanish State Research Agency through the Mara de Maeztu Seal of Excellence to IRI (MDM-2016-0656). Gerard Canal is also supported by the Spanish Ministry of Education, Culture and Sport via a FPU doctoral grant FPU15/00504.

*Both authors contributed equally to this work.

[1]G. Canal, G. Alenyà and C. Torras are with Institut de Robòtica i Informàtica Industrial, CSIC-UPC, C/ Llorens i Artigas 4-6, 08028 Barcelona, Spain. {gcanal,galenya,torras}@iri.upc.edu

[2]E. Pignat and S. Calinon are with the Idiap Research Institute, Martigny, Switzerland. {emmanuel.pignat, sylvain.calinon}@idiap.ch

Therefore, adaptive interaction is a key element for the success and acceptance of assistive robots. Some of the challenges that need to be solved are how to transfer such skills to robots in an easy manner, and how to enable robots to cope with user reactions and other issues that may happen during the task, ensuring a robust and safe behavior. In this paper, we tackle both issues in a joint way. First of all, we use a learning by kinesthetic demonstration approach to teach the robot "smart" low-level movement primitives, so that it can track the user state and move accordingly. Then, a stochastic symbolic planner is used to obtain the sequence of actions to complete the task. If, instead, the high-level task planner were used without the low-level primitives, a higher action granularity and more implementation effort would be needed, and tackling the whole problem with the low-level primitives without task planning would require a larger number of demonstrations. Thus, the main advantages of the proposed joint approach are a reduced number of easier demonstrations, and less symbolic actions with better error handling and robustness.

As an example, we propose the shoe-fitting scenario, where a robotic arm has to put a shoe on a user's foot. We devise different actions for the robot, such as: shoe grasping, approaching the foot, fitting the shoe, and releasing it. Though simple, the task involves physical contact with the user's foot, which may be harmful. Accordingly, the robot must take this into account, know how the user may behave and suitably adapt its actions in order to fulfill the task successfully, recovering from inappropriate situations when needed.

II. RELATED WORK

Service robots in general, and the assistive ones specifically, must perform complex tasks with many particularities. Therefore, joining a high-level symbolic task planner with appropriate low-level motion primitives simplifies the task.

There are many works in literature that address task and motion planning, in a consecutive or integrated way, but most of them focus on the manipulation of still objects, while in our case, we are physically interacting with a person that may move freely.

Gravot *et al.* [1] present a collaborative cooking task with a robot in which a symbolic HTN planner uses cooking recipes to guide the performance of the task and decomposes them into primitive actions, which may be sensing, making specific movements, planning motion or interaction.

Other authors address motion planning as a geometrical problem. De Silva *et al.* [2] propose an interleaved interface

to perform symbolic task planning and geometric planning. The geometric planner is used to compute possible grasps and object locations, taking into account geometric constraints. The symbolic planner is an HTN, and symbolic tasks are related to their Geometric Task Planner's tasks, for which they propose an interleaved backtracking algorithm. The authors apply it in the context of pick-and-place tasks, including human handovers. In a similar manner, Srivastava *et al.* [3] first compute a task plan using a symbolic planner, and then search the instantiations of the pose references used in the plan by means of a motion planner. When such instantiations are not found, partial solutions are identified and extended using the task planner. Another example is the one by Ferrer-Mestres *et al.* [4], where they integrate task and motion planning together, addressing the symbolic and geometrical components of the task simultaneously. Furthermore, Bidot *et al.* [5] present an hybrid task-and-motion planning approach in which task planning is coupled with motion planning and geometric reasoning. Lee *et al.* [6] combine probabilistic activity grammars with low-level motion primitives to learn tasks with reusable structures.

The adaptive component is essential to solve our kind of tasks in an efficient manner, as well as the use of learning by demonstration to simplify the teaching of the movements. Assistive dressing, the problem we are focusing on in this work, has also received some attention from the community. Gao *et al.* [7], [8] assist a user to put on a sleeveless jacket by modeling the user's movement space using a Gaussian Mixture Model so the robot can dress the user taking into consideration his/her movement capabilities. Then, they perform online path optimization to personalize the dressing. A jacket dressing is studied by Chance *et al.* [9]. An IMU sensor installed in the robot's end-effector is used along with a force sensor to explore dressing error detection. Yamazaki *et al.* [10] use visual and force sensing to help disabled users to put on trousers in an adaptive manner. Another example is the collaborative dressing scenario presented by Klee *et al.* [11]. They use a turn-taking approach to move the robot taking into account the user's constraints and learning them. The method is used to put on a hat. These works present interesting approaches to model the user space and capabilities, but lack the ability to demonstrate the task in an easy manner that produces smoother movements and shows a nice adaptability to the user movements.

III. SYMBOLIC TASK PLANNING: OBTAINING THE NEXT ACTION TO PERFORM

Physical Human-Robot Interaction tasks, such as the ones commented above, are appropriate to be tackled from a task planning point of view. Since the robot is in contact with a user, single reactive behaviors may not be enough to deal with user actions in the long-term horizon of the task. Therefore, it is important to have a plan of the robot's actions that should be performed, solving plan deviations as soon as they occur.

A task planning problem $\Sigma = \langle S, A, T, s_0, g \rangle$ is defined by the set of discrete states S, the set A of actions that modify the state, the state transition function T, the initial state $s_0 \in S$ and the goal state $g \in S$. The planner aims to find an action sequence that starts from s_0 and modifies the state using actions in A to end up in g. The state is described by a set of logic predicates, and each action $a_i \in A; a_i = \{p_{a_i}, e_{a_i}\}$ is composed by the preconditions $p_{a_i} \in S$, predicates that must be true in order to perform the action, and the effects $e_{a_i} \in S$, which define how the state changes after the execution of the action. Such state is obtained from the internal (known) robot state, but also from a visual system tracking both the user and other objects related to the task.

Since HRI domains are highly non-deterministic as the user is not a controlled agent, the computed plan should consider unexpected effects. For this reason, we rely on a stochastic representation, which allows the definition of a domain in which actions can yield stochastic effects. In this case, the actions' effects e_{a_i} are not just a unique set but many possible sets of outcomes with an associated probability π_j of happening:

$$e_{a_i} = \begin{cases} \pi_1 : e^1_{a_i} \\ \pi_2 : e^2_{a_i} \\ \vdots \\ \pi_n : e^n_{a_i} \end{cases} .$$

Formally, the problem is usually represented as a Markov Decision Process (MDP) $\langle S, A, P, R, \gamma \rangle$, where $P(s'|s, a)$ defines the probability of going from state s to s' when performing action a, and $R : S \times A \rightarrow \mathbb{R}$ is the reward function associating a score to each action. Then, the planner is set up to find an action sequence that maximizes the reward (or, equivalently, minimizes the cost), while taking into account the probability of each action's outcomes. An application example of this type of planning applied to robotized surface cleaning is provided in [12].

Therefore, we can define the actions so that their possible outcomes are based on user reactions, and the planner will compute a plan by considering the probabilities of each effect. As a result, actions that may produce inconvenient outcomes will be less likely to be selected.

Each symbolic action a_i corresponds to one low-level movement primitive (i.e. physical action). These movement primitives, detailed in Section IV, are self-adapting motions that can track entities of interest for the task, such as the foot in a shoe-fitting scenario.

Once the plan $P = [a_i, a_j, \ldots, a_k]$ has been computed, each action is sequentially carried out. However, there may be cases in which an action produces a non-satisfactory outcome. In such cases, replanning is needed in order to find a new sequence of actions P that brings from the current system's state to the goal state g, and the new plan will be executed.

When interacting with human users, and more specifically when assisting them in a physical manner, it is important to interact with the user, and make clear what the robot is doing. For this reason, we define two interaction actions: inform and

ask. The **inform** action is used to provide verbal information to the person before the execution of each movement to avoid misunderstandings or unexpected situations due to the user misinformation about the robot behavior. The **ask** action is used to obtain user's collaboration in cases in which the task cannot be completed. For instance, in the shoe-fitting scenario, the robot would ask the user to put back the foot, when he/she has moved it to an unreachable position, or to avoid moving it when trying to perform a physical contact.

Not less important while interacting with human users, is to adapt to the specific user the robot is assisting. There are no two equal individuals, so the robot should not assist all the users in the same way, but adapt to their preferences and needs. Following our taxonomy of user preferences in assistive scenarios [13], in the current work we consider two kinds of preferences: the velocity of movements, and the verbosity level.

Performing the task too slowly may result in user fatigue, and doing it too fast may scare the person. Similarly, a too verbose robot may irritate the user, and a non-informative robot may confuse and scare the user. To cope with these dilemmas, we have defined three speed levels $\alpha \in \{slow,\ medium,\ quick\}$, and two verbosity levels $\beta \in \{informative,\ \neg informative\}$. A user model $u = \{\alpha_u, \beta_u\}$ has been added to the planner by means of the predicates α and β.Each action's reward R_i depends on this user model, penalizing such actions that violate it:

$$R_i = R_d - P_\alpha(\alpha_i \neq \alpha_u) - P_\beta(\beta \neq \beta_u), \qquad (1)$$

where R_d is the default action's reward, P_α is the penalty for not following the user's model speed α_u in the current action a_i executed with speed α_i. Similarly, P_β is the penalty for a wrong verbosity β_u. This way, the planner not only computes a plan to solve the task, but also does it while satisfying the user preferences.

Furthermore, the penalties applied when user model violations occur can be also modified in order to favor exploration outside of the current user model in cases of failure, coping with poorly classified users and converging to an appropriate robot behavior. This, however, is out of the scope of this paper.

With the explained approach, the robot is able to compute a plan from s_0 to g that complies with the user model. Moreover, when an unexpected behavior arises, the plan is recomputed and its execution is resumed from the new state. This means that the system is able to recover from errors, repeating previous actions when needed in order to return to a previous state or even starting over the task if required. For instance, in a case in which the shoe-fitting is incorrect, the robot would re-grasp the shoe and start the task if far from the foot, or retry the insertion if the foot is close enough.

IV. LEARNED MOTIONS: MOVING THE ROBOT TO FULFILL THE ACTION

The spatio-temporal definition of a unit action is encapsulated in a statistical model used for time-series, a hidden semi-Markov model (HSMM) [14]. The parameters of this model are learnt by maximizing its likelihood on demonstration data, examples of the action. The same approach was used in [15] to encode the task of putting on a coat.

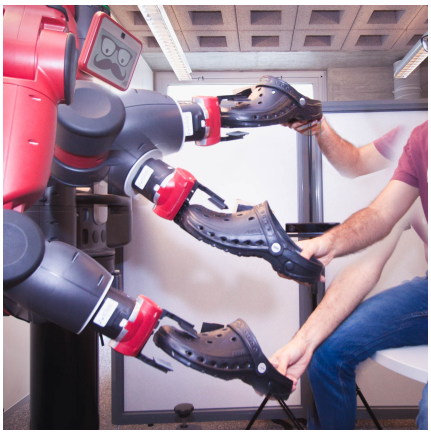

Fig. 1: By providing demonstrations of a task in several situations, the robot is able to generalize to a wide range of new situations.

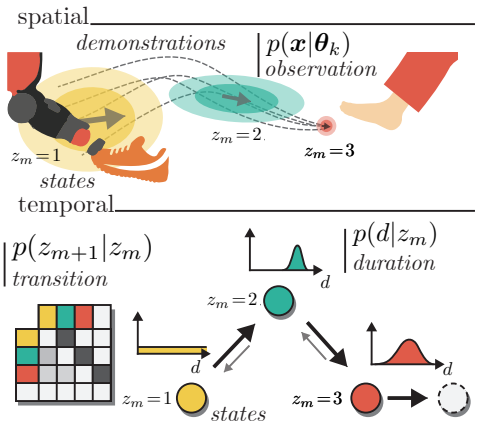

Fig. 2: Example of hidden semi-Markov model encoding the spatio-temporal characteristics of the action of approaching the shoe to the foot. *(top)* A set of continuous distributions, named *states*, encodes positions, velocities, orientations, etc. Each *state* is linked to a *duration distribution*. A *transition distribution* indicates the possible sequencing of the *states*.

A. Temporal representation of movements

The HSMM decomposes a complex skill in a discrete sequence of spatial distributions encoding positions, orientations, velocities, etc. To these discrete distributions, named *states*, are associated a *duration distribution*, indicating how many time steps the state lasts. A *transition matrix*, as in standard hidden Markov models (HMM), encodes the probability of transitioning to any other states, once the current state reaches the end of its duration. These parameters are illustrated in Fig. 2. In robotics, the advantages of HSMM over HMM is the explicit encoding of the duration which allows for a more accurate synthesis of motion subject to precise timing constraints, as pointed out in [16]. Moreover, in our application, modulating this distribution allows the

system to adapt the speed of the actions, which can be determined by a higher-level planner according to the user's preferences.

B. Spatial representation of movements

The problem of adapting the actions to a varying environment is often seen as a problem of regression [17], [18]. In our case, the input of the regression would be the position and orientation of the foot of the user, or the shoe, in the hand-over action, as shown in Fig. 1. If these techniques are general with respect to the nature of the input variable, they do not fully exploit the structure of the problem, thus limiting the adaptation and generalization capabilities, as well as data efficiency. As proposed in [19], the environment will be defined by several coordinate systems in which the data will be projected.

The features encoded in the model of the action are $\tilde{\boldsymbol{y}}_t = \begin{bmatrix}\boldsymbol{x}_t^\top & \dot{\boldsymbol{x}}_t^\top & \boldsymbol{q}_t^\top\end{bmatrix}^\top$, where $\boldsymbol{x}_t$ is the position of the end-effector, $\dot{\boldsymbol{x}}_t$ its velocity and $\boldsymbol{q}_t$ its orientation in quaternion form. The distribution of the data $\boldsymbol{y}_t$, given the current cluster z_t, is the product of normal distributions of the data under P different transformations $\boldsymbol{\mathcal{T}}_t^j$, varying at each time step t

$$p(\boldsymbol{y}_t|z_t) = \prod_{j=1}^{P} \mathcal{N}(\boldsymbol{\mathcal{T}}_t^j(\tilde{\boldsymbol{y}}_t)|\boldsymbol{\mu}_{z_t}^j, \boldsymbol{\Sigma}_{z_t}^j). \tag{2}$$

This can be interpreted as different experts observing the actions from various points of view. In this work, the experts view the data projected in several coordinate systems that are linked to the objects of interest, namely the robot itself, the foot of the user and the shoe. The transformed data is

$$\boldsymbol{\mathcal{T}}_t^j(\tilde{\boldsymbol{y}}_t) = \begin{bmatrix}(\boldsymbol{A}_{t,j}^{-1}(\boldsymbol{x}_t - \boldsymbol{b}_{t,j}))^\top & (\boldsymbol{A}_{t,j}^{-1}(\dot{\boldsymbol{x}}_t))^\top & (\boldsymbol{Q}_{t,j}^{-1}(\boldsymbol{q}_t))^\top\end{bmatrix}^\top$$

where $\boldsymbol{A}_{t,j}$ is the rotation matrix of expert j at time step t, $\boldsymbol{b}_{t,j}$ its position and $\boldsymbol{Q}_{t,j}$ its matrix representation of quaternion form.

The set of experts forms an over-complete representation of the action. The adaptation comes from the combination of their views, some experts encoding more precisely some part of the action (see Fig. 3). As defined in (2), the combination is achieved through a product of experts, which can be evaluated in closed form [20], given that linear transformations of a normal random variable are also normal.

C. Controller

Given a statistical model of the action, the motion is synthesized using optimal control. The density function acts as a cost; the robot tries to track the zone of higher probabilities, both in terms of positions, velocities and orientations. An additional term is included for minimizing the control commands $\boldsymbol{u}_t$, thus providing smooth motion

$$c = \sum_{t=1}^{T} (-\log(p(\boldsymbol{y_t}|z_t)) \; + \; \boldsymbol{u}_t^\top \boldsymbol{R}_t \, \boldsymbol{u}_t), \tag{3}$$

As the $-\log(p(\boldsymbol{y_t}|z_t)$ is quadratic in $\boldsymbol{y_t}$ and given a linearised model of the robot as $\boldsymbol{y}_{t+1} = \boldsymbol{A}_t\boldsymbol{y}_t + \boldsymbol{B}_t\boldsymbol{u}_t$,

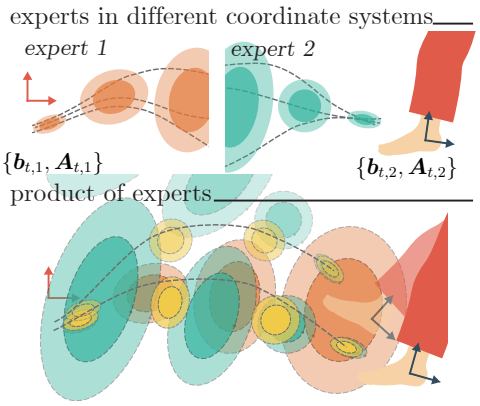

Fig. 3: In order to provide data-efficient adaptation to varying poses of objects, the model (in red and green) is encoded in different coordinate systems, linked to various objects of interest. This can be interpreted as several experts observing the data under different projections. The combination of their knowledge provides the desired adaptation (in yellow).

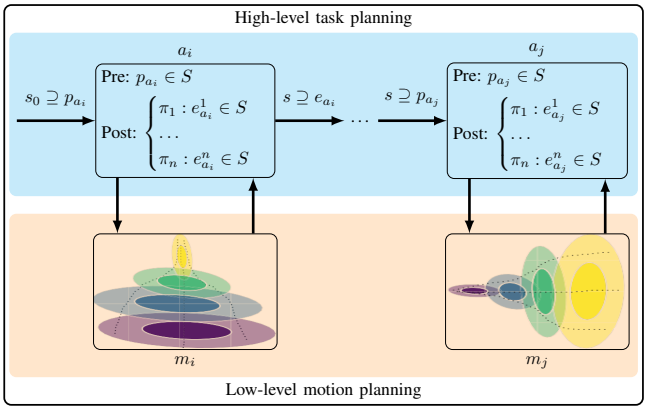

Fig. 4: Two-level planning architecture.

the problem of computing the sequence of input $\boldsymbol{u}_t$ that minimize c reduces to a *linear quadratic tracking* (LQT) problem, which can be solved efficiently.

In this work, the robot is approximated as a rigid body, attached to its end-effector, and of varying inertia. *Operational space control* with *Jacobian transpose* is then used.

V. COMBINING HIGH-LEVEL SYMBOLIC TASK PLANNING WITH LOW-LEVEL MOTION PLANNING

In the previous sections we have defined both system levels and given insights on how they are related. The proposed architecture is depicted in Fig. 4. The symbolic planner's actions have a direct correspondence with the low-level motion primitives. The high-level planner computes the sequence of actions. Then, the low-level primitives, previously learned by demonstration, are executed.

The strength of this approach is to overcome the limitations of using the two levels separately. Although the same task could be tackled from both points of view, the necessary efforts are highly reduced by the union of both.

On the one hand, performing the full task with the sensorimotor motion primitives would require several demonstrations of all the possible events that may happen throughout the task. This results in the need of designing such demonstrations in a thorough manner, taking into account every case in the scenario.

On the other hand, using a symbolic planner to perform the same task would require to split each of the actions in subactions, obtaining a finer granularity. This not only introduces an overhead in the plan computation (as the domain will have many more actions and outcomes), but also requires the design of the domain such that all the possible outcomes of each action are properly defined. Moreover, this also needs us to implement the control and movements of each action in the robot, be it by demonstration or with another technique.

Therefore, by using both approaches together, we obtain a simpler and more efficient planning domain which is easier to design, implement and debug. Such approach requires only few demonstrations of the full task. Typically, the demonstrations take the form of simple atomic movements instead of complete movements. Moreover, such structure facilitates the addition of verbal interactions and the handling of errors by means of replanning. It provides a way to link high-level actions to low-level control commands, facilitating the modulation of low-level actions and the gradual acquisition of complex skills.

Given that the learned motions are adaptable and they do not depend on fixed start and end positions, they are always chainable provided that the high-level actions are also chainable.

A. *High-level state transitions: a shoe fitting example*

In order to demonstrate the advantages of the architecture, we show in Fig. 5 some of the action transitions that are possible in the shoe-fitting task. As it is clearly seen in the graph, the structure of the transitions between actions is quite complex, despite the task requires a low number of actions. Therefore, teaching all the possible transitions by demonstration would result in a tiresome—if not unfeasible—work due to the large number of demonstrations that would be needed. Moreover, the use of the planner permits an easy inclusion of interactive actions such as informing and asking the user, which would have been much harder using a different approach. Thus, the use of the task planner in the high-level loop allows to reuse simple low-level movement primitives, and diminishes the number of demonstrations to just a few demonstrations for each high-level action.

VI. EXPERIMENTAL EVALUATION

The proposed two-level architecture has been implemented using a Rethink Robotics' Baxter robot, in a shoe-fitting application. This section reviews some of the experiments, more details and a visual demonstration of them can be seen in the supplementary video material[1].

[1] www.iri.upc.edu/groups/perception/twoLevelDressing

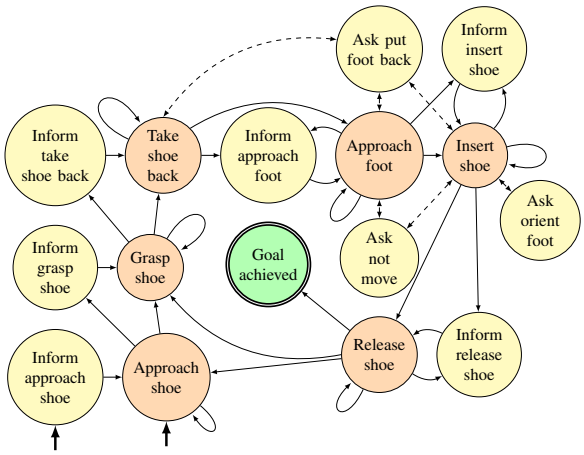

Fig. 5: Example of action transitions in the shoe-fitting task. Orange nodes represent robot motion actions, while yellow nodes represent interaction actions.

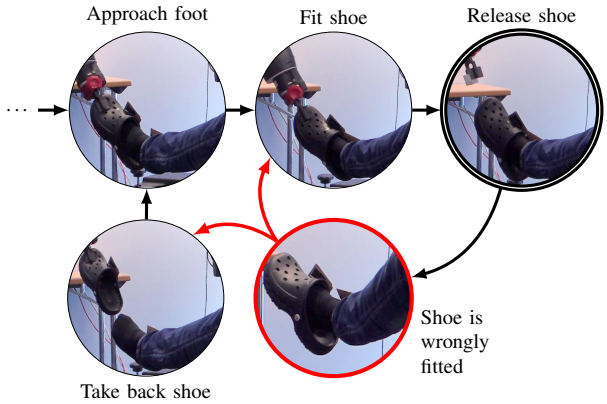

Fig. 6: Example of error state management with the high-level task planner in a non-demonstrated situation.

A. *Experiment 1: Failure recovery after task completion*

To demonstrate how the proposed approach is able to cope with non-demonstrated events, we show an experiment in which the user removes the correctly fitted shoe from the foot, as shown in Fig. 6. The user removes the shoe after the fitting, resulting in an incorrect fit (red node). The planner is able to detect the situation, and recompute a plan to solve the task by grasping the shoe again. Then, the planner decides to fit the shoe if the foot is still close, or to take it back and approach the foot again in case the user has moved the foot away. Notice that all the high-level actions and low-level primitives used in this scenario are the same ones that were implemented for the original task. For instance, the grasp shoe is used either to get the shoe from the user or to pick it up from the foot after the bad positioning of the shoe. Several kinesthetic demonstrations of the situation would have been needed to obtain the same behavior without the high-level task planner.

B. *Experiment 2: Talking to the user when needed*

Here we show how the interaction works. As already introduced, the robot has two interactive actions: ask and inform.

Fig. 7: Example of user moving the foot. This will trigger an *ask* action to stop the movement, and may also produce a speed change.

In a normal execution, the robot may be completely silent, or inform the user if it is defined in the user model. The interesting part is when the execution does not go as expected. When fitting the shoe, the user may become tired or scared, resulting in him/her removing the foot from the robot's working space. Then, in order to complete the task, the robot must *ask* the user to put the foot back in the fitting space, or it may *ask* him/her to reduce the motion of the foot for a correct and safe fit, as shown in Fig. 7. Another case may be one where an action keeps failing because of different user reactions. In such case, the robot *informs* the user about the actions it is performing. With this, the robot may gain user's trust and avoid misunderstandings during completion of the task, even when the model of the user does not take speech into account.

C. Experiment 3: Speed modulation

As already stated, there are many reasons to modify the speed of the actions, among other aspects of the task. In a first case, the robot makes a sudden quick movement that scares the user, as seen in Fig. 7. In this situation, the user performs a reflex action that moves away the foot, preventing the task to be accomplished. In another case, the robot moves too slowly resulting in fatigue that causes the user to remove the foot from the robot's working space. Such cases are tackled by the high-level planner, which takes into account the speed of the actions and the user model to compute a scaling parameter for the low-level primitives.

VII. CONCLUSIONS

In this paper, we have presented a two-level approach to develop assistive robotics applications. In the high-level, a task planner computes a sequence of actions that will bring the robot to the completion of the task. The low-level reproduces the actions, which are learnt by demonstration beforehand, in such a way that they are able to adapt to the current situation by tracking entities of interest for the task.

The proposed approach reduces the number of demonstrations that are needed to implement the task in the robot, and makes them simpler and easier to teach. It also lowers the number of symbolic actions that are needed, as well as diminishes the number of replanning attempts. Moreover, it provides better error handling, resulting in a robust execution of the task.

The high-level stochastic planner and the low-level hidden semi-Markov model (HSMM) are lightly coupled in this work. In the future, we plan to further integrate both components so that probabilities are shared by the high-level and the low-level actions, in order to facilitate the learning of both levels at the same time.

REFERENCES

[1] F. Gravot, A. Haneda, K. Okada, and M. Inaba, "Cooking for humanoid robot, a task that needs symbolic and geometric reasonings," in *IEEE Intl. Conf. on Robotics and Automation*, 2006, pp. 462–467.

[2] L. de Silva, A. K. Pandey, and R. Alami, "An interface for interleaved symbolic-geometric planning and backtracking," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2013, pp. 232–239.

[3] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 639–646.

[4] J. Ferrer-Mestres, G. Frances, and H. Geffner, "Planning with state constraints and its application to combined task and motion planning," in *Workshop on Planning and Robotics (PLANROB)*, 2015, pp. 13–22.

[5] J. Bidot, L. Karlsson, F. Lagriffoul, and A. Saffiotti, "Geometric backtracking for combined task and motion planning in robotic systems," *Artificial Intelligence*, vol. 247, pp. 229 – 265, 2017.

[6] K. Lee, Y. Su, T.-K. Kim, and Y. Demiris, "A syntactic approach to robot imitation learning using probabilistic activity grammars," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1323–1334, 2013.

[7] Y. Gao, H. J. Chang, and Y. Demiris, "User modelling for personalised dressing assistance by humanoid robots," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 1840–1845.

[8] ——, "Iterative path optimisation for personalised dressing assistance using vision and force information," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 4398–4403.

[9] G. Chance, A. Jevtić, P. Caleb-Solly, and S. Dogramadzi, "A quantitative analysis of dressing dynamics for robotic dressing assistance," *Frontiers in Robotics and AI*, vol. 4, 2017.

[10] K. Yamazaki, R. Oya, K. Nagahama, K. Okada, and M. Inaba, "Bottom dressing by a life-sized humanoid robot provided failure detection and recovery functions," in *IEEE/SICE Intl. Symp. on System Integration*, 2014, pp. 564–570.

[11] S. D. Klee, B. Q. Ferreira, R. Silva, J. P. Costeira, F. S. Melo, and M. Veloso, "Personalized assistance for dressing users," in *Intl. Conf. on Social Robotics (ICSR)*. Springer, 2015, pp. 359–369.

[12] D. Martínez, G. Alenya, and C. Torras, "Planning robot manipulation to clean planar surfaces," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 23–32, 2015.

[13] G. Canal, G. Alenyà, and C. Torras, "A taxonomy of preferences for physically assistive robots," in *IEEE Intl. Symp. on Robot and Human Interactive Communication (RO-MAN)*, 2017, pp. 292–297.

[14] K. P. Murphy, "Hidden semi-markov models (HSMMs)," *Technical Report*, 2002. [Online]. Available: www.ai.mit.edu/murphyk

[15] E. Pignat and S. Calinon, "Learning adaptive dressing assistance from human demonstration," *Robotics and Autonomous Systems*, vol. 93, pp. 61–75, July 2017.

[16] S. Calinon, A. Pistillo, and D. G. Caldwell, "Encoding the time and space constraints of a task in explicit-duration hidden Markov model," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September 2011, pp. 3413–3418.

[17] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Rob. Auton. Syst.*, vol. 60, no. 10, pp. 1327–1339, 2012.

[18] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Auton. Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[19] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intell. Serv. Robot.*, vol. 9, no. 1, pp. 1–29, 2016.

[20] C. Williams, F. V. Agakov, and S. N. Felderhof, "Products of Gaussians," *Adv. Neural Inf. Process. Syst. 14*, no. 1, pp. 1017–1024, 2002.