Personalized Assistance for Dressing Users

Steven D. Klee[1], Beatriz Quintino Ferreira[2], Rui Silva[3], João Paulo Costeira[2], Francisco S. Melo[3], and Manuela Veloso[1]

[1]Computer Science Department, Carnegie Mellon University
sdklee@cmu.edu, mmv@cs.cmu.edu
[2]ISR, Instituto Superior Técnico, Universidade de Lisboa, Portugal
{beatrizquintino,jpc}@isr.ist.utl.pt
[3]INESC-ID and Instituto Superior Técnico, Universidade de Lisboa, Portugal
rui.teixeira.silva@ist.utl.pt, fmelo@inesc-id.pt

Abstract. In this paper, we present an approach for a robot to provide personalized assistance for dressing a user. In particular, given a dressing task, our approach finds a solution involving manipulator motions and also user repositioning requests. Specifically, the solution allows the robot and user to take turns moving in the same space and is cognizant of the user's limitations. To accomplish this, a vision module monitors the human's motion, determines if he is following the repositioning requests, and infers mobility limitations when he cannot. The learned constraints are used during future dressing episodes to personalize the repositioning requests. Our contributions include a turn-taking approach to human-robot coordination for the dressing problem and a vision module capable of learning user limitations. After presenting the technical details of our approach, we provide an evaluation with a Baxter manipulator.

Keywords: Human-Robot Interaction, Dressing, Human Tracking, Learning and Adaptive Systems

1 Introduction

Research has moved towards allowing robots and humans to operate in the same workspace. For instance, human-aware robots can assist people in factories, or even elderly people in their homes. In particular, there has been increasing interest in developing robots that can help people overcome their disabilities and limitations [13]. There are many challenges involved in designing robotic systems that deliver highly personalized assistance to different individuals, owing to the uncertainty introduced by human presence.

In this paper, we address the problem of providing personalized assistance to help dress a person with a manipulator. To this end, we introduce a framework for human-robot collaboration, in which the user and robot take turns moving to complete their shared goal. During these interactions, the robot learns the user's limitations and uses this information to provide personalized interactions.

Dressing tasks, represented as templates, are sequences of goal poses with respect to the user. For instance, placing a hat on a user's head has one goal pose, with the arm several centimeters above his head. Other tasks may have several goal poses that must be incrementally solved to complete a task.

Before the robot acts on these templates, they are instantiated with the current location of the user. For example, for a 1.8m tall person, a goal position of 10cm above the head of the user becomes 1.9m from the ground. This is the first form of personalization which allows templates of sequential goals to be parameterized by the user's physical features.

Once the template is instantiated, the robot attempts to fulfill the goals with its motion planner, while asking the user to remain still. If the plan fails, the robot tries to re-instantiate the template by asking the user to move to reposition himself. Specifically, a planner determines a sequence of user repositioning requests that will take the user from their current pose to a new pose.

As a second form of personalization, the robot models each user's limitations, represented as pose constraints. When the robot asks the user to reposition himself, it selects a pose that satisfies these constraints. Specifically, the robot selects a new pose for the user by sampling points in its configuration space that also satisfy the known constraints. Constraints are learned by a vision module that monitors the user's response to repositioning requests. Each repositioning request has an expected behavior, and the robot infers a new constraint when the expectation is not met. Due to inaccuracies in vision, and the uncertainty introduced by humans, these constraints have error tolerances and confidences. To increase our confidence and refine constraints, they are ignored in future episodes with probability proportional to their confidence.

Among our contributions, the three most notable are:

- An approach to the dressing problem that explicitly requests the user to reposition himself when a solution cannot be found.
- A vision-based expectation algorithm that determines if the user is complying with an interaction request and learns his constraints.
- A turn-taking approach for safe human-robot interactions in a shared workspace.

In the remainder of the paper, we compare our approach to related works, in Section 2. Then, in Section 3, we provide an overview of the proposed approach before detailing each component. In Section 4 we describe a specific implementation with a Baxter manipulator robot and show experimental results. Finally, we draw conclusions and discuss future work in Section 5.

2 Related Work

Our work is similar to research in designing algorithms for manipulators that help dress humans. Much prior work has focused on the problem of clothing manipulation in simulation [3] and also in the real world [7,9,14]. Our work is more focused on the additional challenges posed when a person is involved.

When operating a manipulator near a person, accurate human tracking, localization, and fusing information from available sensor networks is extremely important [12]. Researchers have developed approaches for dressing people based on visual and sensory information [15]. Their proposed visual framework matches features acquired from depth and color images to features stored in a database. Similar to our approach, this system can detect failures via vision. However,

on failures we re-plan using a series of user interactions and motion planning, whereas they repeat the same action assuming the error was transient.

Researchers have used reinforcement learning to teach robots to put on a t-shirt and wrap a scarf around a mannequin's neck [10,5]. In the latter work, the robot learns tasks through dynamic movement primitives, allowing them to modify the trajectory speed or goal location. By contrast, our approach uses a sampling-based motion planner. By performing motion planning online, we have a high degree of confidence that the trajectory will not collide with the person.

The aforementioned works do not attempt to *cooperate* with the user. Our approach employs *sparse-coordination*, where the robot asks the user to reposition himself/herself if it cannot otherwise solve the task [11]. Additionally, similar to the concept of maintenance goals, repositioning requests have expectations that are monitored over time by a vision module [2]. Lastly, our work draws from research in representing and solving generalized or parameterized plans [6]. In particular, we represent dressing tasks as sequences of subgoals that are parameterized by the location and size of the user.

3 Approach to Personalized Dressing

Our aim is to enable a manipulator to aid users in dressing tasks by providing personalized assistance. Throughout this paper, we employ the example of helping a user to put on a hat. Figure 1 shows our Baxter manipulator ready to perform this task.

Fig. 1: Baxter equipped to dress a user with two different hats.

Our approach is described at a high-level in Algorithm 1. In particular, each dressing task is a *template*, which is a sequence of goal poses of the manipulator with respect to the user. The *template* is instantiated with the current position and orientation of the user by the vision module (Line 3). Then, the motion planner attempts to find and execute a motion plan that satisfies each goal (Line 4). If it meets a goal that is infeasible, the robot attempts to re-instantiate the template by choosing a new pose for the user that satisfies his known constraints and is in the robot's configuration space (Line 6). To refine constraints, some are ignored with probability proportional to their confidence. The robot then determines a sequence of repositioning requests that will move the user to the chosen pose (Line 7). Finally, the vision module monitors the user during repositioning requests and infers new limitations when he cannot respond (Line 8). This process happens in a loop until all of the template's goals are solved.

Algorithm 1 Execution of Personalized Human-Robot Motion Interactions

```
1: procedure EXECUTE(template, vision, motion, constraints)
2:     while goals ≠ ∅ do
3:         goals ← vision.instantiate(template)
4:         goals ← manipulation(goals, motion) // remaining goals
5:         if goals ≠ ∅ then
6:             newPose ← feasiblePosition(template, vision, constraints, motion)
7:             interactions ← planInteractions(vision, constraints, newPose)
8:             constraints ← interaction(interactions, vision, constraints)
```

3.1 Vision-Based User Tracking

The vision module monitors the user to pause motion execution if he moves, and to determine if he is complying with repositioning requests.

The vision module is provided with a set of joint positions J from a skeleton tracker. We model a person as a set of connected body parts, B:

$$B = \{head, torso, left\_arm, right\_arm\}$$

that are tracked by the vision module, which provides the center-of-mass location (x, y, z) and orientation $\langle q_x, q_y, q_z, q_w \rangle$ of each body part. Typically $|J| > |B|$ and some joint locations map directly to body part locations. The body part orientations are computed by looking at the locations of consecutive joints.

Our vision module then performs two additional functions:

- Detecting if a person is stopped or moving.
- Verifying user-repositioning expectations.

To detect if a person is stopped or moving, the robot compares the body part positions and orientations over time. If $b_t \in B$ is a body part at time t, the body part is stopped if:

$$\forall k \in [t - \gamma, t] : ||b_k - b_t|| < \epsilon$$

Where γ is the time the user must be stopped for, and ϵ is a threshold to allow for small movements or tremors.

When the robot asks the user to reposition himself, it generates expected poses for some of the user's body parts and sends them to the vision module. When the user next stops moving, vision verifies that the user's actual body part poses match the expected poses. The verification of whether the user is in an expected position is done by the function *verify_expectation*(*expected_pose*, *actual_pose*), which compares the expected pose of each user's body part to the actual poses found by the vision. It returns *True* if each dimension matches up to a configurable threshold and *False* otherwise. When the result of this function is *False*, the robot infers a new user constraint or refines an existing one.

3.2 Dressing Tasks as Template Goals

We represent a dressing task as a sequence of desired manipulator pose goals with respect to the user. For instance, putting on a hat has one desired pose, with the arm straight and above the user's head. Other tasks may have several

desired poses that must be completed incrementally. For instance, putting on a backpack has several goal poses to help the user get each arm into a strap. These sequences of goals are general to any user, but are parameterized based on their physical characteristics. Formally, a *template goal* T is a sequence of pose goals:

$$T = \langle P_1, P_2, P_3 ...., P_n \rangle$$

Where each pose goal is composed of a position (x, y, z), orientation as a quaternion $\langle q_x, q_y, q_z, q_w \rangle$, and a vector of tolerances for each dimension (TOL):

$$P_i = \langle x, y, z, q_x, q_y, q_z, q_w, TOL \rangle$$

Thus, P_i is a goal pose $\langle x, y, z, q_x, q_y, q_z, q_w \rangle$, which is satisfied by a pose P^* if:

$$\forall d \in P_i \quad |P_{i_d} - P^*_d| \leq TOL_d$$

Goal poses have allowable tolerances in each dimension for two reasons. First, they account for minor vision inaccuracies. Secondly, some tasks may not require exact orientations or positions, as the user can readjust himself.

A template is *instantiated* when the vision module provides the location of a user with respect to the manipulator. Then, each pose P_i is transformed from the reference frame of the user to the reference frame of the robot. This allows the robot to run a motion planner for each pose goal.

3.3 Motion Planning

We adopt a sparse-interaction approach, where the manipulator first tries to solve as many pose goals as it can, before asking the user to reposition himself. To minimize the possibility of a collision the manipulator only moves when the user is stopped. If the user begins moving, the robot halts and re-plans once the user stops. The vision module is responsible for monitoring the user.

The algorithm for our approach is shown in Algorithm 2. Given instantiated template *goals*, the motion planner attempts to incrementally solve and execute the pose goals. The manipulator represents the world as a 3D occupancy grid provided by the vision module. We then use a sampling-based motion planner to compute a trajectory through the joint-space to our goal position. Specifically, we use RRT-Connect, a variant of Rapidly-Exploring Random Trees (RRT) known to work well with manipulators [8]. RRT-Connect grows two trees and attempts to connect them by extending the two closest branches. If the manipulator finds a pose goal that is infeasible, the robot chooses a new location for the user and determines a sequence of repositioning requests to ask.

3.4 User-Aware Pose Selection and Assistance Planning

Once the manipulator determines that a pose goal is infeasible, it asks the user to reposition himself. The robot has a knowledge-base of user constraints and balances seeking feasible poses with refining the constraints. Once a new pose is determined, the robot finds a sequence of interaction requests that will move the user to this pose. Each interaction is monitored by a vision system that infers new constraints when the user cannot comply with a request.

Algorithm 2 Motion Planning

```
1: procedure MANIPULATION(goals, motion, vision)
2:     for P_i ∈ goals do
3:         plan ← motion.rrt(P_i)
4:         if plan == ('Infeasible') then
5:             break
6:         while plan.executing() do
7:             if vision.feedback() == 'User Moving' then
8:                 motion.pauseUntilUserStopped(vision)
9:                 plan.replan(vision)
10:    return goals.remaining()
```

We represent a constraint c on body part $b \in B$ as:

$$c = \langle b, ineq, conf \rangle$$

Where *ineq* is an inequality for a limitation on one of the pose dimensions. The left hand term of the inequality is the dimension, while the right hand term is the value that cannot be surpassed. For instance, $torso.x > 0.9$ means that the torso cannot move closer than 0.9m to the base of the robot. $conf \in \mathbb{R}$, represents the robot's confidence in the constraint. This is a function of how many times the constraint is satisfied compared to the number of times it is violated.

Given a set of known feasible manipulator poses $\mathcal{P}$, future poses are sampled from points in $\mathcal{P}$ that satisfy all *active* user constraints. For each interaction, a user constraint is *active* with probability proportional to its confidence. In this way, the robot can test less confident constraints to refine them.

Once a new pose is found, the robot must determine a sequence of interactions that will reposition the person. Each body part has a position, orientation, and set of *motion actions*. The motion actions represent parameterized repositioning requests that the robot can ask of the person. We define several translational motion and rotational actions including: $forwards(x)$, $backwards(x)$, $left(x)$, $right(x)$, $up(x)$, $down(x)$, $turn\_right(\theta)$, and $turn\_left(\theta)$. Where x is a distance in meters and θ is a rotation in degrees with respect to the manipulator.

Each of these motion actions is associated with a set of expectations E:

$$e = \langle b, L, TOL \rangle$$

Where each expectation consists of a body part $b \in B$, expected pose L, and tolerance on each dimension of the pose TOL. For instance, requesting $forwards(1)$ on the torso corresponds to asking the user to walk 1m forwards and moves all of the body parts. By contrast, requesting $up(0.2)$ on the left arm will only move the left arm up 0.2m, because it corresponds to lifting an arm.

Using these expectations, we can generate a plan of translational and rotational user requests that repositions each of their body parts. In particular, we focus on repositioning the user to the correct x position, then y, then z, and finally, the correct rotation. More generally, a planner could pick the order of these actions. During these interactions, the user is monitored by the vision

module which determines if he complies with a request. If the user cannot, the robot infers a constraint, selects a new pose, and tries again.

3.5 Learning and Refining User Constraints

The robot learns and refines constraints in two situations. The first is when a user stops before reaching an expected pose during a repositioning request. The second is when the user enters a space the robot believed to be constrained.

In the first case, given a motion action m and an expectation $\langle b, L, TOL \rangle$, the vision module detects that the user has stopped, and is not in the expected position. Then, a constraint is inferred for the body part b on the axis associated with m. The constraint is that the user can move no further than his current position, which is provided by the vision. If a constraint along this axis already exists for that body part, it is updated with the new value.

For instance, the action $forwards(x)$ expects all of the body parts to move x meters closer to the robot. If the user cannot comply with this request a constraint c_b is generated for each body part $b \in B$ of the form:

$$c_b = \langle b, b.x > vision.b.x, conf \rangle$$

where $vision.b.x$ is the x-coordinate of body part b, and $conf$ is a configurable initial confidence value.

In the second case, the user moves into a space that the robot believed to be constrained. This implies that the original constraint was too strict, and should be relaxed. To do this, the robot replaces the value in the inequality with the value from the user's current location.

In both cases, the robot updates the confidence in the constraint. This influences how often the constraint is ignored when picking a user-pose to request.

Let N_s be the number of times the user approaches a constraint and does not pass it (the constraint is satisfied); and N_f the number of times the constraint fails, because the user passes it. The confidence $conf$ of a constraint is:

$$conf = \frac{N_s}{N_s + N_f}$$

This constraint inferring procedure becomes particularly relevant when interacting with disabled people. For instance, a user in a wheelchair may not be able to move perfectly laterally, or too close to the robot. In such cases, learning constraints in the adequate axes will optimize future interactions, as the robot will request poses that satisfy the inferred constraints. We note that the constraints learned are approximations of a human's physical limitations, as joint movements are dependent from one another.

4 Evaluation

We tested our approach with a Baxter manipulator. Baxter is a particularly good platform because is designed to work alongside users, implementing force-compliant motion. We mounted a Microsoft Kinect on Baxter, and used OpenNI-Tracker to find human joint positions. This depth image based skeleton tracking approach produces robust joint positions for different users in various background

and lighting conditions [1]. Baxter makes repositioning requests by displaying text on its screen.

For these experiments, we considered the task of putting on a hat on the user's head. Formally, the goal template $T = \langle P_1 \rangle$ for this task has one goal pose. In this pose, the right gripper of Baxter points straight out, positioned 10cm above the *head* body part.

We tested our approach with users ablating themselves to simulated physical limitations. In the first set of experiments, we show how increasing the constraint complexity affects the execution time of the task. This motivates our constraint learning approach. In the second set of experiments, we show that learning the constraint model of a user optimizes the interactions. In particular, with enough accurate constraints the execution time is similar to the case with no constraints.

4.1 Planning and Executing with Increasing Constraint Complexity

We tested our system in the real world with one taller person (1.83m tall) and one shorter person (1.62m tall), who ablated themselves by simulating physical limitations. Our aim was to show that users with more complex constraints take longer to teach, which motivates learning personalized constraint models. Our first constraint was that the user was in a wheelchair and could not move his head closer than 0.9m to the robot. The second was that the user could not move his head more than 0.2m left.

We consider four cases, with results shown in Table 1. In Column A, the user starts in a feasible pose, and the manipulator immediately executes its task. In Column B, the robot selects a user-feasible pose, the user repositions himself, and the manipulator is then able to execute the task. In Column C, the user has one constraint, and the robot initially selects a user-infeasible pose. Thus the robot first makes an infeasible user-repositioning request and must detect this. Then, it must sample a new pose outside the user constraint space, and repeat the repositioning process. In Column D, the user has two constraints, so the process from Column C is likely to occur multiple times. We note that the execution time is highly dependent on the number of instructions the robot gives the user, and on how promptly the user complies with them.

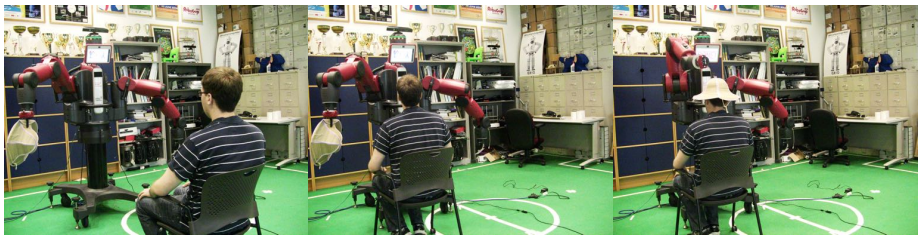
	A	B	C	D
Execution time (s)	11.6 ± 1.8	32.0 ± 12.3	53.0	78.0
Number of interactions	0	2.1	3.3	3.5

Table 1: Average execution times and interactions for the 4 cases over 40 trials.

The variances for Columns C and D are omitted as there was a significant discrepancy between the two users. The distribution was bimodal, implying that the two users took different times to respond to the robot's instructions.

Fig. 2 shows a task execution for a user constrained to a chair, meant to model a wheelchair. Initially, the template goal is instantiated by the vision module with the user's head location. At first, the user is too far away, and the motion planner cannot solve the goal. The robot selects a new pose by sampling points in its configuration space that are believed to be feasible for the user. Additionally, the robot determines that this pose be achieved by asking the user

to move forwards. Then, the vision system detects when he halts in the expected position and re-instantiates the template goal. Finally, the motion planner finds this new goal to be feasible, solves it, and places the hat on the user's head.



(a) User is initially at an unreachable pose.

(b) Robot requests the user to move forward.

(c) Manipulator successfully puts hat on the user.

Fig. 2: Example of dressing task execution for a user constrained to a chair.

The large majority of trials succeeded in putting a hat on the user, with few errors. Occasionally, the motion planner failed when the user was positioned at extremes of the reachability space of the manipulator, which can be resolved by considering these as invalid poses. As for the vision module, there were some problems when the robot arms occluded the field of view, which could be mitigated by adding additional cameras, using the built-in cameras on Baxter's arms, or planning trajectories to avoid obstructing the camera. As we will show in the next Subsection, after several trials of interaction with the same user to learn his specific constraints, future dressing episodes become more efficient.

4.2 Learning User Models

In this second set of experiments we show that, resorting to the knowledge-base formed by previous interactions with each user, the robot is able to complete the task more efficiently. To demonstrate this, we used the set of 2 constraints defined in the preceding subsection: $C = \langle head, head.x > 0.9, conf\rangle, \langle head, head.y > -0.2, conf\rangle$, where our starting confidence value *conf* was 1.

We first taught Baxter a perfect constraint model for each user. Then, we performed 5 additional trials per user, with the robot choosing repositioning poses from the users non-constrained space. The average time for completing the task was $(34.2 \pm 29.4)s$ for the shorter user and (36.5 ± 8.5)s for the taller user with an average of 2.6 interactions. These results are very close to those for a user with no constraints (Table 1 Column B), which shows that learning personalized constraint models can markedly optimize the task execution.

5 Conclusion

In conclusion, we introduced an approach for a manipulator to help dress users. Our approach represents a dressing task as a sequence of pose goals with respect to the user. When the robot finds a goal infeasible, it actively requests the user to reposition his or herself. The robot chooses repositioning poses by sampling points in its configuration space that also satisfy a user-constraint model. This

constraint model is updated when the user does not meet the expectations of an interaction request or enters a region the robot believes to be constrained.

We demonstrated our approach on a Baxter manipulator. We first showed how the time it takes to dress a user increases with constraint complexity. Then, we showed how modeling a user's constraints and sampling from feasible regions reduces the dressing time close to the case of a user with no constraints.

One direction for future work is to improve the manipulator's trajectories to make them more user-friendly. This could possibly be accomplished by biasing the motion planner with user-taught trajectories such as with E-RRT [4].

Acknowledgements

This research was partially supported by a research donation from Google, by NSF award number NSF IIS-1012733, by the FCT INSIDE ERI grant and two FCT student visiting fellowships. The views and conclusions contained in this document are those of the authors only.

References

1. Anjum, M.L., Ahmad, O., Rosa, S., Yin, J., Bona, B.: Skeleton Tracking Based Complex Human Activity Recognition Using Kinect Camera. In: ICSR (2014)
2. Bacchus, F., Kabanza, F.: Planning for temporally extended goals. Annals of Mathematics and Artificial Intelligence 22(1-2), 5–27 (1998)
3. Bai, Y., Liu, C.K.: Coupling cloth and rigid bodies for dexterous manipulation. In: Motion in Games (2014)
4. Bruce, J., Veloso, M.: Real-time randomized path planning for robot navigation. In: IROS (2002)
5. Colomé, A., Planells, A., Torras, C.: A Friction-model-based Framework for Reinforcement Learning of Robotic Tasks in Non-rigid Environments. In: ICRA (2015)
6. Fikes, R.E., Hart, P.E., Nilsson, N.J.: Learning and executing generalized robot plans. Artificial intelligence 3, 251–288 (1972)
7. Kita, Y., Neo, E., Ueshiba, T., Kita, N.: Clothes handling using visual recognition in cooperation with actions. In: IROS (2010)
8. Kuffner, J.J., LaValle, S.M.: Rrt-connect: An efficient approach to single-query path planning. In: ICRA (2000)
9. Maitin-Shepard, J., Cusumano-Towner, M., Lei, J., Abbeel, P.: Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In: ICRA (2010)
10. Matsubara, T., Shinohara, D., Kidode, M.: Reinforcement learning of a motor skill for wearing a t-shirt using topology coordinates. Advanced Robotics 27(7), 513–524 (2013)
11. Melo, F.S., Veloso, M.: Decentralized mdps with sparse interactions. Artificial Intelligence 175(11), 1757–1789 (2011)
12. Quintino Ferreira, B., Gomes, J., Costeira, J.P.: A unified approach for hybrid source localization based on ranges and video. In: ICASSP (2015)
13. Tapus, A., Mataric, M., Scasselati, B.: Socially assistive robotics [grand challenges of robotics]. Robotics Automation Magazine, IEEE 14(1), 35–42 (March 2007)
14. Twardon, L., Ritter, H.J.: Interaction skills for a coat-check robot: Identifying and handling the boundary components of clothes. In: ICRA (2015)
15. Yamazaki, K., Oya, R., Nagahama, K., Okada, K., Inaba, M.: Bottom dressing by a life-sized humanoid robot provided failure detection and recovery functions. In: System Integration (2014)