Trajectory Adaptation of Robot Arms for Head-pose Dependent Assistive Tasks

Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni
Department of Computer Science
University of Central Florida
{rrahmati, pabolghasemi, lboloni}@eecs.ucf.edu

Amirhossein Jabalameli, Aman Behal
Department of Electrical and Computer Engineering
University of Central Florida
amir.jabal@knights.ucf.edu, abehal@ucf.edu

Abstract

Assistive robots promise to increase the autonomy of disabled or elderly people by facilitating the performance of Activities of Daily Living (ADLs). Learning from Demonstration (LfD) has emerged as one of the most promising approaches for teaching robots tasks that are difficult to formalize. LfD learns by requiring the operator to demonstrate one or several times the execution of the task on the given hardware. Unfortunately, many ADLs such as personal grooming, feeding or reading depend on the head pose of the assisted human. Trajectories learned using LfD would become useless or dangerous if applied naïvely in a situation with a different head pose. In this paper we propose and experimentally validate a method to adapt the trajectories learned using LfD to the current head pose (position and orientation) and movement of the head of the assisted user.

Introduction

With the extension of the human lifespan and general aging of the population in many advanced countries it becomes increasingly important that elderly and disabled people are empowered to live autonomously. Assistive robots, which can be either wheelchair mounted robotic arms or mobile robots with one or more manipulators can help disabled people perform the Activities of Daily Living (ADLs). Many ADLs depend on the head pose of the user. Examples include feeding (using forks, spoons, glasses, bringing bottles of juice or medication to the mouth), personal grooming (combing the hair, shaving, brushing teeth) as well as other ADLs such as reading a book or participating in a video chat. This means that the assistive robot cannot simply reproduce pre-programmed or rigidly learned trajectories. The trajectory must be dependent on and adapting to the current head pose of the user. For instance, when feeding the user, the robot needs to bring the fork to the mouth of the user – other trajectories are ineffective and potentially dangerous.

Adapting to the current head pose is not a problem if the trajectory had been calculated from scratch using a formal model of the task. Unfortunately, many ADLs are difficult to describe formally - they might depend on the environment and the preferences of the user. The most desirable way to teach a robot to perform an ADL is Learning from Demonstration (LfD) - a technique in which the task is demonstrated to the robot either by manually guiding its arm or by teleoperation. Ideally, from a small number (possibly, just one) demonstration, the robot should be able to generalize the learned trajectories to a new environment or state. Due to the increase in the problem complexity, previous LfD implementations that adapt the demonstrated trajectories to new situations usually do not generalize to the 3D pose of the objects in the environment.

Related work

In some problems, gathering relatively large number of examples is possible in the simulation environment(e.g. (Abolghasemi et al. 2016)). However, the challenge in many LfD applications is to use a minimum number of examples to learn a task. The examples may include situations in which the geometry of the objects change from the demonstration scene to the test scene (Schulman et al. 2013), where non-rigid registration is used to map the camera input points from the training scene onto the test scene. This registration is later used to adapt the trajectory of the robot arm captured during the demonstration to the testing situation. It turns out that non-rigid registration works well as long as the environment does not change too much such that a valid registration could not be found. For example, when person's head rotates $180°$, most of the points of the demonstration are not visible anymore, hence, finding a warping function fails.

Another method to capture the critical aspects of demonstrated trajectories and maintain them during the test is investigated in (Ye and Alterovitz 2011). Their method handles new positions of the involved objects while avoiding new obstacles in the scene by using motion planning algorithms. A method that utilizes averaging to generalize the trajectory to the scenes where the object position is changed is proposed by (Reiner et al. 2014). Neither method considers the orientation of the objects, thus they cannot be applied to situations where a non-symmetric object rotates.

(Pastor et al. 2009) extend the Dynamic Movement Primitives framework in which recorded movements can be represented with a set of differential equations. They adapt this framework to generate trajectories that end at goals at different positions while avoiding obstacles. (Calinon et al. 2009) use a Hidden Markov Model to capture the constraints of demonstrated trajectory. They use a set of pre-defined landmarks for each trajectory to track the changes from demonstration to test.

More recently, (Jain et al. 2013) worked on improving the demonstrated trajectories based on user preferences. Finally, (Rozo, Jiménez, and Torras 2013) proposed a LfD framework teaching force-based manipulation tasks to robots.

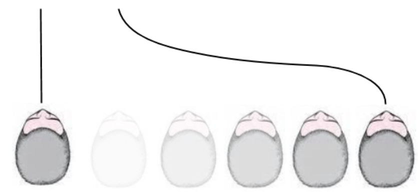
Problem definition

In learning from demonstration, we start by performing the task for the robot and recording the executed trajectory. A task can be executed using different trajectories based on the state of the environment. For example, a robot performing a head pose dependent ADL task might use either its right or left arm depending on which one is closer to the head. Therefore, we record trajectories augmented with the state of the environment at every time step. For the purpose of the work described in this paper, the state will be composed of only the head pose. However, a complete sequence of RGB-D frames from the camera can be considered as the state if we have enough example demonstrations.

The trajectory of the robot arm is usually represented in *joint space* by keeping track of the value of each joint at each time step. This trajectory can also be shown in *task space* in which the pose of the end-effector is stored. We use task space trajectory in order to be able to transform it in 3D space.

For each task, we record N demonstrations $D = \{d_1 \dots d_i \dots d_N\}$. A demonstration $d_i = \{E, Q\}$ consists of Q, the state of environment including head pose H, and $E = [e_1 \dots e_t \dots e_T]$ a set of end-effector poses e_t at time $t = [1 \dots T]$. Pose $e_t = [X, Y, Z, \phi, \alpha, \psi]$ is the vector containing the position and orientation (roll, pitch, yaw) of the end-effector with respect to origin. Similarly, we show the set of head poses during the demonstration by $H = [h_1 \dots h_t \dots h_T]$ in which h_t is the head pose at time t.

Naturally, the robot is able to retrace a learned trajectory, which would work fine provided that the evolution of the head poses during test are the same as during the demonstration. The challenge we are trying to solve is that the head pose during test time is different from the demonstration ones: $H' = [h_1 \dots h_{t'} \dots h_{T'}]$ where $h_{t'}$ is the head pose at time t'. The goal is to find the corresponding end-effector trajectory during the test time $E' = [e_1 \dots e_{t'} \dots e_{T'}]$. Note that not only the head poses might be different, but also the total time of the trajectory t' might be different at test time than the time t it took at demonstration. For example, consider the scenario shown in Figure 1 in which during the demonstration the arm traverses a straight trajectory towards the head. During the test, however, the trajectory takes longer to be executed since the head moves simultaneously. This change in trajectory execution time might occur because we prioritize maintaining the end-effector pose relative to the head pose rather than progressing in execution of the trajectory.



(a) (b)

Figure 1: The duration of executed trajectory at test might be different from the demonstration. Demonstrated trajectory (a) is a straight trajectory towards the head. The desired trajectory at test time (b) is longer since the head moves during the execution.

Trajectory transfer method

In this section, we explain the steps to adapt the trajectories from the demonstration to the test situation. The real-time head pose of the user is collected using a Kinect sensor mounted on the robot. We use this information to transfer each waypoint of a demonstrated trajectory to make a new trajectory. The input of this transformation is the 3D pose of the end-effector, so the result would be a 3D trajectory of the end-effector. Therefore, in order for the robot to be able to execute the trajectory, we convert the trajectory from task space to joint space using inverse kinematics.

Let us start by defining the notations used in the remaining of this section. Each pose in a 3D world can be uniquely described by its position and orientation. We define operators $p(x)$ and $r(x)$ which decompose pose x into a translation vector and a rotation matrix respectively. We also define $p(\Delta(x, y)) = p(x) - p(y)$ to be a translation vector from x to y and $r(\Delta(x, y)) = r^{-1}(y)r(x)$ will be the difference between two rotation matrices.

Finding changes in the head pose

The head pose is extracted using the random regression forests method proposed by (Fanelli et al. 2013). In practice, we found the output of this method to be relatively noisy. By taking advantage of the fact that we are recording a continuous scene at fixed intervals, we applied an Exponentially Moving Average (EMA), a common noise reduction technique for time-series data:

$$h_t = \alpha \tilde{h_t} + (1 - \alpha) h_{t-1} \tag{1}$$

where $\tilde{h_t}$ is the noisy head pose, and h_t is the filtered head pose by considering previous head poses with more emphasis on the most recent ones. The discount factor α controls how much weight we give to the old data, which is set to 0.2 in our experiments.

Each demonstration consists of multiple trajectories. At each time step, we need to select from the demonstrated trajectories one that is “closest” to the test situation. For this purpose, we use a K-Nearest Neighbor (KNN) classifier to decide which head pose in the demonstrated trajectories is closest to the current head pose at this time step. Based on this prediction, we select a waypoint of the corresponding

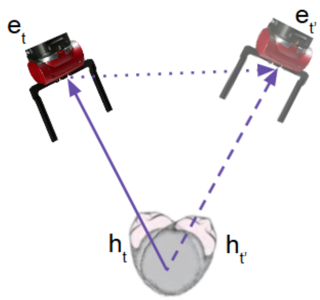


Figure 2: Illustration of equation 4. The solid vector is $p(\Delta(h_t, e_t))$, the dashed vector is $r(\Delta(h_{t'}, h_t))p(\Delta(h_t, e_t))$ and the dotted vector is $p_{rot}(\Delta(e_{t'}, e_t))$.

trajectory at the current time step to be translated. The distance measure used in KNN is as follows:

$$d(h_t, h_{t'}) = \|p(\Delta(h_t, h_{t'}))\| + \beta\|r(\Delta(h_t, h_{t'}))\| \quad (2)$$

where $h_{t'}$ is the head pose at test time step t', h_t is the head pose at demonstration time step t, and the parameter β controls how much weight we give to the orientation compared to the position. Note that the selection of the trajectory occurs at each time step during the execution of the trajectory. In other words, at each time step, we consider the head pose in each trajectory and select the closest one, then we go to the next time step.

Transferring end-effector poses

The objective in this part is to calculate $p(e_{t'})$ and $r(e_{t'})$ which is position and orientation of robot's end-effector at test time t' based on $p(h_t)$, $r(h_t)$, $p(h_{t'})$, $r(h_{t'})$, $p(e_t)$ and $r(e_t)$. The assumption is that the end-effector should maintain its previous pose with respect to the head. To simplify the problem, let us divide the changes in the pose of the head to changes in its position and changes in its orientation. If the head only moves without any change in its orientation, the desired pose for the end-effector can be achieved by the same head translation:

$$p_{trans}(\Delta(e_{t'}, e_t)) = p(\Delta(h_t, h_{t'})) \quad (3)$$

On the other hand, if the head only rotates and does not move as shown in Figure 2, the end-effector's corresponding action will be a translation to keep the same position with respect to the head:

$$p_{rot}(\Delta(e_{t'}, e_t)) = r(\Delta(h_{t'}, h_t))p(\Delta(h_t, e_t)) - p(\Delta(h_t, e_t)) \quad (4)$$

and a rotation to adjust the orientation.

$$r_{rot}(\Delta(e_{t'}, e_t)) = r(\Delta(h_{t'}, h_t))r(e_t) \quad (5)$$

Now we can calculate the overall translation and rotation matrix from e_t to $e_{t'}$ as

$$p_{overall}(\Delta(e_{t'}, e_t)) = p_{trans}(\Delta(e_{t'}, e_t)) + p_{rot}(\Delta(e_{t'}, e_t)) \quad (6)$$

$$r_{overall}(\Delta(e_{t'}, e_t)) = r_{rot}(\Delta(e_{t'}, e_t)) \quad (7)$$

The calculations we have performed up to this point are finding the new pose of a single waypoint in a trajectory. In practice, however, we need to transform the whole trajectory so that it can perform the same task with respect to the new pose of the head. The first observation we make is that not all the points in the trajectory are required to be transformed equally: for instance, points closer to the head require should be translated more compared to the points far away from the head. To decide how much a point in a trajectory should be transformed, we use the following logistic function as a Translation Factor (TF) for each waypoint based on their distance to the head:

$$TF(d) = \frac{1}{1 + e^{k(d - \hat{d})}} \quad (8)$$

where $d = p(\Delta(e_t, h_t))$ is the euclidean distance between the end-effector and the center of the head, $\hat{d}$ is the midpoint of transformation, and k is the slope of the transformation. The translation can vary from 0 when the end-effector is far enough from the head to 1 when the end-effector is near the head. The parameters $\hat{d}$ and k can be decided based on the task.

Finally, we can achieve the new position and orientation of the end-effector using these formulas:

$$p(e_{t'}) = p(e_t) + p_{overall}(\Delta(e_{t'}, e_t)) \times TF(d) \quad (9)$$

$$r(e_{t'}) = r(e_t) + r_{overall}(\Delta(e_{t'}, e_t)) \times TF(d) \quad (10)$$

Converting end-effector trajectory to joint angles

In order for the robotic arm to be able to execute the trajectory, a joint configuration should be found such that the end-effector reaches the desired pose. This can be achieved using inverse kinematics. We assume that the demonstration used a smooth movement to perform the task. It is possible, however, that the trajectory resulting from the transfer is not going to be smooth, because, as shown in Figure 1, the change in the head pose might insert new trajectory sequences. If the robot tries to make these corrections too quickly in an attempt to keep to the original schedule, it can result in a jerky movement. To mitigate this problem, before transferring the trajectory from task-space to joint-space, we interpolate between the successive end-effector poses which are far from each other in 3D space. Then, we transfer the trajectory to the joint space. We also designed a mechanism to control the speed of the arm by interpolating between successive joint configuration waypoints. This control is implemented at the joint level just before the command is sent to the robot to make sure that the trajectory is smooth and safe.

Experiments

We have implemented our technique using the Baxter robot by Rethink Robotics. Baxter has a zero-force gravity compensation mode in which a user can steer the robot's arm to desired configurations. While the user is moving the arm, we record the joint configurations and also the pose of the end-effector at a frequency of 20Hz. This series of recorded

Figure 3: Sequence of images demonstrating the task of holding a book for the user to read. Notice that as the the user turns his head, the robot positions the book for a comfortable reading position at an appropriate reading distance.

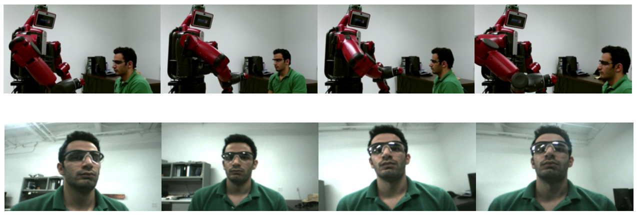


Figure 4: Sequence of images demonstrating the execution of the tasks of recording a video of subject's face for facilitating a video chat. The top row shows the relative position of the user and the Baxter robot, while the bottom row shows the video captured. Notice that although the user had moved around significantly, his face remains centered in the video stream.

end-effector poses augmented with the gripper status forms the trajectory of the arm.

For our experiments we considered three different ADLs:

- Holding a book for the user such that he can read it.
- Facilitating a video chat by recording the face of the user using a camera mounted on the wrist of the robot.
- Bringing a bottle of water close to the user's head.

The experiments had been performed as follows. A human subject sits in front of the robot in such a way that the robot arm can reach his head. A Microsoft Kinect sensor mounted on the Baxter tracks the head pose of the user by capturing RGB-D frames. Based on the relative pose of the Baxter and the Kinect with respect to each other in the real world, we use a translation matrix to convert the points from Kinect coordinate system to the Robot's coordinate system. The captured frames are processed in real-time and the extracted head pose is recorded at the same rate as recording end-effector trajectory waypoints.

In the task of recording video from the person's face, the camera on the Baxter's arm is used. In this task, the head of the person is centered to the camera frame. We expected that by using our proposed method, the head should remain in center when the person moves or rotates his head.

Results and Limitations

The sequence of images in Figure 3 shows how the robot performs the tasks of holding a book for the subject while Figure 4 shows the task of facilitating a video chat. In addition, the video of the robot executing the tasks can be found online[1]. In the tasks of holding a book and recording video, our algorithm could successfully adapt the demonstrated trajectory in real-time as the human subject was moving.

For normal operating conditions we have found that the robot was able to achieve all the three tasks successfully. We have, however, also identified some limitations of the trained trajectories.

In the task of bringing a bottle of water close to the human head, the translation factor TF plays an important role. If we set the parameter so that it is a large number even for points far from the head, the rate of unsuccessful trials will increase. On the other hand, if we tune it so that even the points close to the head are not translated completely, the bottle will not end up in a proper position close the the subject's head. Therefore, a trade-off needs to be made to tune this parameter.

Acknowledgments: This work had been supported by the National Science Foundation under Grant Number IIS-1409823.

References

Abolghasemi, P.; Rahmatizadeh, R.; Behal, A.; and Bölöni, L. 2016. A real-time technique for positioning a wheelchair-mounted robotic arm for household manipulation tasks. In *Workshop on artificial intelligence applied to assistive technologies and smart environments (ATSE-16) at AAAI*.

Calinon, S.; D'halluin, F.; Caldwell, D. G.; and Billard, A. G. 2009. Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 582–588.

Fanelli, G.; Dantone, M.; Gall, J.; Fossati, A.; and Van Gool, L. 2013. Random forests for real time 3d face analysis. *International Journal of Computer Vision* 101(3):437–458.

Jain, A.; Wojcik, B.; Joachims, T.; and Saxena, A. 2013. Learning trajectory preferences for manipulators via iterative improvement. In *Neural Information Processing Systems (NIPS)*, 575–583.

Pastor, P.; Hoffmann, H.; Asfour, T.; and Schaal, S. 2009. Learning and generalization of motor skills by learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 763–768.

Reiner, B.; Ertel, W.; Posenauer, H.; and Schneider, M. 2014. Lat: A simple learning from demonstration method. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4436–4441.

Rozo, L.; Jiménez, P.; and Torras, C. 2013. A robot learning from demonstration framework to perform force-based manipulation tasks. *Intelligent Service Robotics* 6(1):33–51.

Schulman, J.; Ho, J.; Lee, C.; and Abbeel, P. 2013. Learning from demonstrations through the use of non-rigid registration. In *International Symposium on Robotics Research (ISRR)*.

Ye, G., and Alterovitz, R. 2011. Demonstration-guided motion planning. In *International Symposium on Robotics Research (ISRR)*, volume 5.

[1] `https://youtu.be/BU775Wdd4JU`