--Preprint--

Shared Dynamic Curves: A Shared-Control Telemanipulation Method for Motor Task Training

Daniel Rakita,[1] Bilge Mutlu,[1] Michael Gleicher,[1] Laura M. Hiatt[2]
[1] Department of Computer Sciences, University of Wisconsin–Madison, Madison, Wisconsin
[2] Naval Research Laboratory, Washington, District of Columbia
{rakita, bilge, gleicher}@cs.wisc.edu; laura.hiatt@nrl.navy.mil

ABSTRACT

In this paper, we present a novel shared-control telemanipulation method that is designed to incrementally improve a user's motor ability. Our method initially corrects for the user's suboptimal control trajectories, gradually giving the user more direct control over a series of training trials as he/she naturally gets more accustomed to the task. Our shared-control method, called *Shared Dynamic Curves*, blends suboptimal user translation and rotation control inputs with known translation and rotation paths needed to complete a task. Shared Dynamic Curves provide a translation and rotation path in space along which the user can easily guide the robot, and this curve can bend and flex in real-time as a dynamical system to pull the user's motion gracefully toward a goal. We show through a user study that Shared Dynamic Curves affords effective motor learning on certain tasks compared to alternative training methods. We discuss our findings in the context of shared control and speculate on how this method could be applied in real-world scenarios such as job training or stroke rehabilitation.

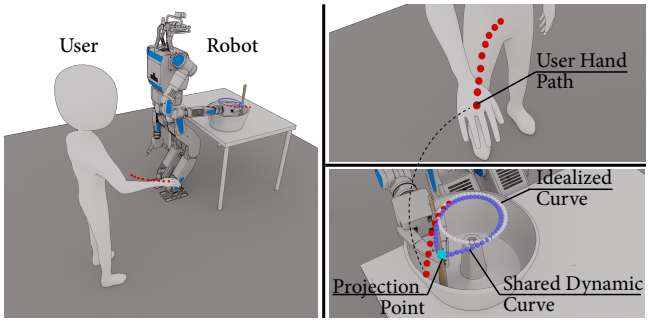
ACM Reference Format:
Daniel Rakita, Bilge Mutlu, Michael Gleicher, Laura M. Hiatt. 2018. Shared Dynamic Curves: A Shared-Control Telemanipulation Method for Motor Task Training. In *HRI '18: 2018 ACM/IEEE International Conference on Human-Robot Interaction, March 5–8, 2018, Chicago, IL, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3171221.3171278

1 INTRODUCTION

In this paper, we explore how shared-control-based methods for telemanipulation of robot arms can adapt to and improve user ability to manipulate objects. Telemanipulation methods that are based on "direct control" reflect all the commands provided by the user, whereas methods for "shared control" enable the robot to take the user's commands as a guideline and blend its own control signals into a single motion [18]. Much of prior research on shared control has investigated *how* to blend between user and robot actions [1, 3, 4, 8, 16, 25], but fewer studies have focused on *when* shared control would be useful in practice [9, 17, 20].

In this work, we consider *motor task training* as a setting in which shared-control-based manipulation might significantly benefit users by shifting the focus of shared control from increasing overall task proficiency to providing effective skill learning for the user. In this context, even if the robot has full understanding of the task and could perform it autonomously or with minimal input, users maintain partial control in order to improve at the task themselves in scenarios such as job training or stroke rehabilitation. For example, a patient recovering from stroke who does not have the muscle strength or dexterity to pick up a cup from the counter but wants to regain the muscle coordination to complete the task over time could start by *guiding* a robot through the task, which could then incrementally relinquish control to the patient until the patient demonstrates the necessary strength and dexterity. Compared to direct-control approaches, which would not provide the necessary assistance for patients to perform tasks, thus "under-helping," and standard shared-control approaches, which would continuously provide assistance and would not facilitate skills learning, thus "over-helping," a shared-control paradigm that enables a therapist to tailor assistance to the changing skill levels of the patient might in the short term provide such patients with the ability to perform tasks as early as possible and in the long term regain motor skills, boost morale, and regain independence [12].

Figure 1: Our shared-control method for motor task training: (1) the user demonstrates a motion with their own hand and arm (right-top); (2) the robot solves for a full curve from start to goal, called a *Shared Dynamic Curve*, that blends an idealized curve with the user's motion (right-bottom).



Our work is rooted in the learning-theory concept of *scaffolding*, a teaching strategy intended to tailor support to a particular learner, then gradually remove the support structure as they begin to discover their own strategies [23]. This method provides users with the opportunity to work through their "zone of proximal development," the gap between tasks that can only be done with assistance and tasks that can be completed independently [2]. The central challenge in this work is to establish the notion of scaffolding in a shared-control setting to a sufficient degree to provide users with the same learning benefits reported from other

domains in the learning-theory literature. To address this challenge, we present solutions to two main questions: (1) *How should the robot provide support through a particular motor task*; and (2) *How should the teacher—the robot—gradually hand over control to the pupil—the user—to foster independent discoveries in motor task strategy?*

We present a novel algorithm that gradually hands over control from the robot to the user over a series of training trials, which is critical for applying the concept of scaffolding to motor task learning. Our algorithm smoothly blends a user's potentially suboptimal input with idealized operational-space curves for translation and rotation that are needed to complete a task. Our method, called *Shared Dynamic Curves*, provides a translation and rotation curve in space along which the user can more easily guide the robot, and this curve can bend and flex in real-time to pull the user's motion gracefully toward the goal (Figure 1).

In the implementation of our method, users perform a motion with their own hand and arm in free-space; a motion-capture system picks up this motion; and the robot reflects the user's arm motion in real-time using the *mimicry-control interface* proposed in prior work [21]. Because the user is controlling the robot with representative motions, the user still gains *experience* in performing the motor task, while the robot maintains the ability to subtly *correct* the user's suboptimal input motions when necessary. We report on a user study that compares *Shared Dynamic Curves* against three baselines: (1) a direct-control approach that *under-helps* users, (2) a shared control approach that *over-helps* users through continuous assistance, and (3) a shared control approach that also gradually relinquishes control to the user but uses *virtual fixtures* [15, 22] to provide guidance. In the study, participants aim to improve at using a direct mimicry-control interface to control a DRC Hubo Humanoid Robot[1] across several telemanipulation tasks that resemble day-to-day object-manipulation tasks.

In the remainder of the paper, we provide motivation and context for our method (§3) and technical details of its implementation (§4), describe the design of and results from the user study (§5), and discuss the implications of our findings as well as the prospects of the use of *Shared Dynamic Curves* in important scenarios such as job training or stroke rehabilitation (§6).

2 RELATED WORK

In this work, we draw from prior work in the robotics literature on shared control. We also look to robotics research in rehabilitation and training to ground our motor training method in prior approaches to motor training.

Shared Control— Shared-control-based teleoperation methods aim to reduce the tedium or difficulty of direct control by letting the robot handle some aspects of the control process, thus unburdening the user from the demands of low-level teleoperation and enabling them to reason about higher-level task objectives [18].

Prior work has explored several different shared-control strategies. One popular strategy involves considering the user's input data as a *state model* to infer their likely goal and giving control over to the robot when a goal is inferred. Javdani et al. [8] refer to this shared control paradigm as *predict-then-act*. Early work in this area considered shared control to be a discrete process in which the robot could switch between predefined levels of assistance [10]. More recent approaches, such as the *policy-blending* approach proposed by Dragan and Srinivasa [4], present ways for the robot to continuously blend the user's and robot's policies on the fly. Dragan and Srinivasa [4] make the important point that the arbitration between robot and user policies should depend on the robot's confidence in its goal inference, presenting a method rooted in inverse reinforcement learning that blends policies based on this key idea. Nikolaidis et al. [19] present a method that automatically decides how the robot should intervene in control based on the user's adaptability. Work by Javdani et al. [8] uses *hindsight optimization* to arbitrate control over a distribution of possible outcomes, even when the robot is unsure about any single particular outcome. Our work is informed by these shared-control methods, although our goal is different, as we aim to facilitate motor task training by guiding along paths to a known goal rather than pursuing inferred goals.

Our method shares more similarities with shared-control methods that involve *potential fields* and *virtual fixtures*. Potential fields are used in teleoperation to guide the user toward a goal point and away from obstacles in the environment [3]. Virtual fixtures allow path following by blending an ideal instantaneous direction and the user's velocity vector at each update [15, 22]. Marayong et al. [16] provide a geometric version of virtual fixtures along multi-dimensional subspaces. This implementation will serve as the comparison baseline in our user study as described in §5. Work by Aarno et al. [1] considers the case of switching between multiple virtual fixtures, such as when completing various sub-tasks, and shows that a classifier can effectively recognize when the user is switching between different task segments. Our work was inspired by this approach for multi-segment tasks, although we used a pre-defined finite state machine rather than a data-driven model.

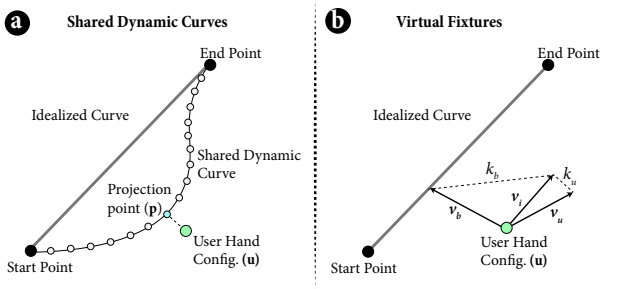
Rehabilitation & Skills Learning— Our method is also inspired by prior research in rehabilitation or skills training. Research in rehabilitation robotics has assessed the benefits of using a robot for stroke rehabilitation [11]. In rehabilitation settings, using haptic guidance training is a common method that uses a robot as a force-resistance device to build strength [5]. Nudehi et al. [20] used a haptic shared-control device to enable an experienced surgeon to guide a novice surgeon through a surgical procedure. Our work similarly uses shared control as a teaching method, although the assistance provided by the robot in our system is fully autonomous rather than being driven by an expert user. Lastly, prior shared-control work has explored how brain signals can be blended with robot policies to allow paralyzed individuals to regain motor abilities by controlling a robot arm through neural activity [9, 17]. Our shared-control method could be used in a similar domain to help train patients to improve their ability to control the robot using similar forms of user input.

3 OVERVIEW OF APPROACH

Our goal in this work is to develop a shared-control telemanipulation method that can teach users how to better perform motor tasks. In this section, we will present the overall premise of our method and discuss key differences between our method and prior shared-control methods that may induce better motor task learning.

[1]Rainbow Robotics: http://www.rainbow-robotics.com

Figure 2: Overviews of (a) our method and (b) virtual fixtures, a common method for shared-control-based path following. Our method solves for a full path from start to goal at each update, rather than updating based on input velocities.



3.1 Overview of *Shared Dynamic Curves*

At a high level, our method involves six key steps. At each system update, the method (1) captures the user's hand position and orientation at the current time, (2) compares the user's hand configuration to a known, idealized way of performing the current task, and (3) calculates a full position and orientation 6D-curve from a start point to a goal point that reconciles the different motor strategies observed in Step 2. The aggressiveness of the pull toward the idealized strategy is dictated by how much assistance is desired during the current training trial. The method then (4) projects the user's current hand configuration onto the curve calculated in Step 3, (5) splits the projection point into position and orientation goals, and (6) calculates new joint values that are then sent to the robot using an inverse-kinematics solver.

Our main technical contribution is at Step 3, specifically how we reconcile the user's potentially suboptimal motion strategy with a known, idealized way of performing the task. We solve for a full 6D curve at each update that mixes between the two strategies. We call this curve a *Shared Dynamic Curve (SDC)*, as it dynamically updates to try to match the *shared policy* between the teacher—the robot—and the pupil—the user—in the context of motor task training. The SDC bends and morphs at each update, and the extent to which the robot assists in the task during training is dictated by how much "flexibility" the SDC has to bend away from the idealized curve, which is controlled by a single parameter. The projection in Step 4 pulls the user's motion down onto the curve, facilitating easy motion along the invisible 6D curve in space. Because the SDC always leads toward the goal, motion toward the goal is always encouraged. An illustration of our method can be seen in Figure 2-a. §4 describes our approach to solving for a full 6D SDC fast enough for use in real-time control.

To accommodate many subtasks within a larger task, such as picking up a pitcher of orange juice to pour into a cup, we chain together multiple SDCs to form separate *task segments*, similar to the approach used by Aarno et al. [1]. However, rather than a data-driven approach, we use simple finite state machines to represent known, sequential paths.

When solving for an SDC at each update, we must have clear objectives for what the curve should try to achieve. We have three main goals for the SDC curve that will foster effective training. (1) The curve should be smooth and continuous, preferably up to the third derivative of position (jerk). Prior work shows that people move their hands along minimum-jerk trajectories when completing tasks [6]. Thus, guiding the robot along similarly smooth paths, even when the robot is providing assistance, will lead to a more natural remapping of the user's motion, which may bolster learning effects. This motion behavior adheres to the important telemanipulation concept of *transparency*, that dictates that control should feel as close as possible to how a user would perform the task themselves without sacrificing stability [13]. (2) The curve should split the difference between the user's current hand configuration, $\mathbf{u}$, and the idealized curve. This requirement means that the projection point, $\mathbf{p}$, will lie somewhere between $\mathbf{u}$ and the idealized curve, which will reflect the user's motion but still adhere to a motion strategy that is known to be effective. (3) The curve at time t should be similar to the previous curve at time $t-1$, which will ensure smooth motions of the projected vector $\mathbf{p}$ over time.

3.2 Comparison to Prior Methods

As mentioned in §2, our *Shared Dynamic Curves* method shares parallels with the *virtual fixtures* approach [1, 16, 22]. The paragraphs below will contrast our method to virtual fixtures, discussing why our method may be better suited for scaffolding motor task learning.

Virtual fixtures is a well established method for guiding users along a pre-defined subspace [22]. A common metaphor for virtual fixtures is using a simulated ruler to improve line drawing, where the ruler is a fixture that encourages motion only along a particular path. An illustration of this overall approach can be seen in Figure 2-b. At a high level, virtual fixtures instantaneously provides an idealized direction $\mathbf{v_i}$ given a current position, linearly blends the user's instantaneous velocity input, $\mathbf{v_u}$, with a certain mixing value, $k_u \in [0, 1]$, and scales the mixed velocity vector to match the user's current velocity. This process can mix in an additional vector, $\mathbf{v_b}$, that drives the user control point back to the idealized curve with mixing value $k_b \in [0, 1]$ (Figure 2-b).

While virtual fixtures have been shown to improve performance over direct control in certain telemanipulation tasks [22], various characteristics of this method could hinder its effectiveness in motor task training. First, because the approach is only loosely tied to positions by vector $\mathbf{v_b}$ and is primarily driven by velocities, it is possible for the user control point to diverge away from the given curve or skate past goal positions, leading to errors and lack of trust in the robot during training. Second, while the mixing value k_u seems straightforward to adjust to gradually relinquish control to the user following the notion of scaffolding, the addition of a grounding vector $\mathbf{v_b}$ makes parameter tuning difficult. Marayong et al. [16] report on choosing k_b through trial and error, finding too high and too low gain values to respectively result in instability and convergence of the tooltip to the reference path slowly. Such unpredictability and the need to tune parameters through trial and error are not well-suited for motor task training, as any instability in robot motion would jeopardize trust in the robot teacher and hinder learning effects. Finally, because the approach updates only based on local, instantaneous information, it is not possible to encourage global features, such as smoothness or continuity of trajectories from a start point to a goal point, which may result in discontinuities in motion. Users may perceive jittery motion to be

too different from how they would complete the task themselves, which may diminish their feelings of gained experience through the task during training.

Our method overcomes issues associated with virtual fixtures by solving for a smooth, full trajectory from a start point to a goal point at each system update. Our solution only considers *positions*, as opposed to velocities, in 6D operational space, thus making it easier to ensure that the user control point will not diverge or skate past the goal. Our method was designed to be easily tunable, specifically amenable for gradually handing over control from the robot to the user. Additionally, because we solve for a full operational space trajectory at each update, global smoothness parameters can be encouraged such that the control behavior feels natural.

4 SHARED DYNAMIC CURVES

In this section, we describe our novel algorithm for generating *Shared Dynamic Curves* at each system update. We address how the particle-flow dynamical system works to encourage desired path objectives, how to accurately accommodate rotations into the SDC, and how to effectively provide scaffolded assistance.

4.1 Particle-flow Dynamical System

A dynamical system is a mathematical framework where the subsequent state of the system is only influenced by the current state along with a rule specifying how items move. In our case, the system is composed of a set of particles and a motion rule that determines how these particles should flow through the 6D operational space. The points are connected as splines at the particle knot points to form a function, which parameterizes our *Shared Dynamic Curve* object, denoted as $\zeta(s) \in \mathbb{R}^6, s \in [0,1]$. The set of n particles making up the curve $\zeta(s)$ is denoted as $\{p_1, p_2, ..., p_n\}$. The 6D position of a particle p_i at time t is specified as $\mu(p_i)_t$.

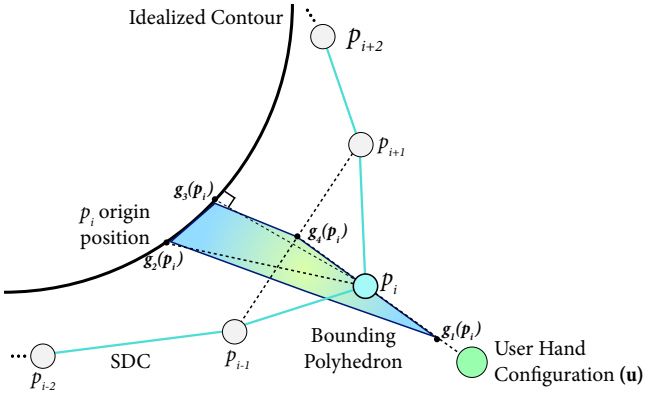
As illustrated in Figure 3, at every system update, each particle position $\mu(p_i)$ is calculated based on the update rule, $\mu(p_i)_{t+1} = \mu(p_i)_t + v(p_i)_t$. Here, $v(p_i)_t$ is a calculated velocity vector for particle p_i at the current time. A magnitude constraint, $||v(p_i)|| < v_{max}$, is enforced to add stability to the system. This constraint enables us to achieve Goal 4 specified in §3.1, because each particle can only deviate so far from its last position.

To calculate the particle velocity vectors $v(p_i)_t$ at each update, we use four *goal points* that attempt to encode the curve objectives specified in §3.1. We refer to the goal points for particle i at time t as $g_1(p_i)_t, g_2(p_i)_t, ..., g_4(p_i)_t$, as shown in Figure 3. We will describe how to solve for each goal point in the following section, §4.2. The four goal points form the vertices of a 6D polyhedron. We consider these polyhedron corner points as barycentric coordinates and create relative importances for each point using scalar weights:

$$g(p_i)_t = \omega_1 g_1(p_i)_t + \omega_2 g_2(p_i)_t + ... + \omega_4 g_4(p_i)_t \quad (1)$$

Here, $g(p_i)_t$ is a single, aggregated goal point formed by mixing the subgoal points using various weights. The ω weights here afford the ability to give relative importances to each subgoal where a higher weight signifies a more important objective for the particle to pursue. Then, $v(p_i)_t$ is calculated by emanating a vector to the goal point from the current particle's position: $v(p_i)_t = g(p_i)_t - \mu(p_i)_t$.

Figure 3: Our *Shared Dynamic Curves* algorithm updates the positions of particles that make up the curve, enabling the SDC to bend and flex in real-time as a dynamical system.



The velocity vector is clipped if the velocity magnitude constraint $||v(p_i)|| < v_{max}$ is surpassed.

The result in Equation 1 can lead to chaotic behavior if weights are not selected carefully. Thus, we use absolute barycentric coordinate weights, i.e., weights such that $\sum_{j=1}^{4} \omega_j = 1$, as $g(p_i)$ will lie within the polyhedron bounding box [24]. This approach leads to desirable, stable behaviors, such as that $\mu(p_i)$ does not drift beyond the user's hand configuration point $\mathbf{u}$, which is possible using virtual fixtures.

At each system update t, the position of each particle is updated using this approach first from start to goal, $\{p_1, ..., p_n\}$, and then from goal to start, $\{p_n, ..., p_1\}$, which ensures that the updated positions are balanced and not skewed, as particle positions depend on their neighboring particles.

4.2 Particle Goal Points

The four goal points that make up the bounding polyhedron vertices, as illustrated in Figure 3, include the following:

(1) $g_1(p_i) = \mu(p_i) + \frac{1}{1+a\phi(p_i)^2}(\mathbf{u} - \mu(p_i))$. Here, $\phi(p_i)$ is the rank index of how close particle p_i is to the user's hand configuration point $\mathbf{u}$ such that the closest particle will be 1, the next closest will be 2, and so on. At a high level, the SDC is pulled toward the user's hand configuration, achieving Goal 2 discussed in §3.1. The quadratic falloff function $\frac{1}{1+a\phi(p_i)^2}$ defines how the particles organize and swim toward the user's hand configuration. This function offers smooth higher derivatives, achieving Goal 1 discussed in §3.1. The a value defines how the quadratic function should drop off and depends on the number of particles in the SDC. In our prototype system described in §5.1, where each SDC curve has approximately 30 particles, we use a value of $a = 0.01$.

(2) $g_2(p_i) = \sigma(p_i)$. Here, $\sigma(p_i)$ signifies the starting location of the particle and discourages the particle from drifting too far from where it started, helping us achieve Goal 2 discussed in §3.1.

(3) $g_3(p_i) = proj_{\gamma(s)}[\mu(p_i)]$. This point is calculated by projecting the particle onto the idealized curve $\gamma(s)$, which pulls the particle back to the idealized curve along the shortest path and encourages the SDC to maintain the shape of the idealized curve as a whole, helping us achieve Goal 2 discussed in §3.1.

(4) $g_4(p_i) = 0.5\mu(p_{i-1}) + 0.5\mu(p_{i+1})$. This position is the midpoint between the neighboring particles. It encourages continuity in curve C^1 and adds to the smoothness of the path, also helping us achieve Goal 1 discussed in §3.1.

4.3 Handling Rotations

Throughout §3 and §4, we have been working with 6D curves consisting of three position components and three rotation components. While the position components are standard x,y,z coordinates in Euclidean space, calculating rotations is not straightforward. In order to correctly represent rotations in a path-following shared-control method, we properly linearize quaternions by taking the logarithm map [14]. The logarithm function of quaternions returns a 3-vector or a *rotation vector*. The particle position updates, interpolations, and any necessary rotation filtering are all safely applied in this space. After processing and updating the positions of particles, the rotation vector components of the 6D curves can be exponentiated back to quaternions using an exponential map. These quaternions can then be passed to an inverse-kinematics (IK) solver to update the robot's joint states. For more detail on representing rotations, we refer the reader to work by Lee [14].

4.4 Tuning Parameters for Scaffolding

Our work aims to enable users to improve at motor tasks by aggressively helping them at first and then gradually relinquishing control to the user over a number of training trials as they get better at the task. To achieve this goal, we designed our method to be easily tunable to gracefully hand control over from the robot (teacher) to the user (pupil). At a high level, scaffolding in *Shared Dynamic Curves* is based on the flexibility of the SDC. This flexibility is controlled by the magnitude of the weight ω_1 relative to the other weights. As specified in §4.2, the $g_1(p_i)$ goal point is what attracts particles toward the user's hand configuration point $\mathbf{u}$. Thus, if ω_1 is low, the SDC will resist flexing and bending away from the idealized curve, and assistance will be aggressive upon projection. Alternatively, if ω_1 is high, more of the user's motion will be reflected as the curve stretches away from the idealized curve toward $\mathbf{u}$. Therefore, we can consider ω_1 as a *scaffolding parameter* and use it as a dial to smoothly control how much assistance the robot is providing through the motor task training trials. We note that all of the weights must sum to 1 such that they form absolute barycentric coordinates, as described in §4.1. Thus, as the ω_1 *scaffolding parameter* is adjusted, ω_2, ω_3 and ω_4 are scaled accordingly.

5 USER STUDY

To test the premise that our shared-control motor training method affords improved learning, we performed a user study involving human participants performing three motor tasks.

5.1 Implementation of the Prototype System

We have realized our *Shared Dynamic Curves* method in a system designed to provide sufficient performance and safety in order to demonstrate its benefits in a user study. The paragraphs below describe details of our prototype system.

Input Device—To capture the user's hand configuration at each system update, we used a Vicon motion-capture setup involving eight cameras placed in a semi-circle around the workspace. The users wore a velcro glove on their right hand with five passive markers secured to it. The marker pattern was recognized as a rigid body, serving as a proxy for the user's hand configuration. The Vicon system provided hand-transform data at a rate of 100 Hz. We expect any motion capture system that can track positions and rotations of the user's hand to support our method.

Robot—The robot platform used in our prototype system was a DRC-Hubo+ humanoid robot from Rainbow Robotics. We controlled only the right arm of the robot. The robot's 7-DOF arm follows an anthropomorphic design and includes a three-finger gripper. The user opened and closed the gripper using a remote.

IK Solver—To solve for robot's joint states at each update, we used the *Relaxed-IK* solver described in previous work [21]. This IK solver is specifically designed to handle numerous objectives on the fly and is especially adept at handling human-to-robot motion retargeting in real-time. The solver was tailored to the Hubo robot in order to avoid self collisions and kinematic singularities.

System Architecture—Our prototype system was set up as a distributed system over a network of computers that utilized ROS for communication. The motion-capture system sent transform information to the SDC-solver computer that solved for the 6D goal point using our *Shared Dynamic Curves* algorithm, which was then sent to a retargeting engine running the *Relaxed-IK* solver, and the robot joint-state solutions were sent to the robot's internal computer to control the joints. This process as a whole ran at approximately 50–60 Hz. The Shared Dynamic Curves algorithm completes an update in approximately 6 ms using 30 particles.

5.2 Hypotheses

Based on prior findings from the learning-theory and robotics literatures, we developed the following hypotheses:

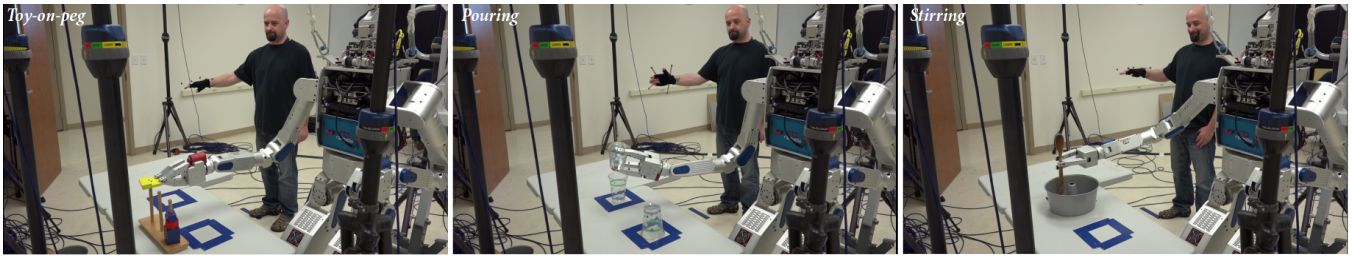
H1—Our principal hypothesis was that our Shared Dynamic Curves training method would lead to individual learning benefits in robot control motor tasks over comparable, state-of-the-art methods. Based on prior work demonstrating the effects of scaffolding and zone of proximal development [2], we expected our SDC method to present a more pronounced improvement in motor skill for users compared to other task training alternatives.

H2—We also hypothesized that a robot controlled using Shared Dynamic Curves would be *perceived* as being more fluent in cooperation, intelligent, and having a better understanding of the user's goals compared to other training approaches, because the sufficient implementation of scaffolding will present a more collaborative and engaging robot teacher. We also expected our method to invoke feelings of more trust in the robot compared to virtual fixtures, because of possible drawbacks, such as skating past the goal, difficult parameter tuning leading to instability, and lack of global smoothness in motion, that our method does not exhibit.

5.3 Experimental Design, Tasks, & Procedure

Our user study followed a 4×1 between-subjects design. The participants used one of four training methods (*Shared Dynamic Curves*, *Direct Control*, *Overhelping*, and *Scaffolded Virtual Fixtures*) to receive training on three tasks (*toy-on-peg*, *pouring*, and *stirring*).

Figure 4: The three motor training tasks used in our study. In randomized order, all participants received training on placing a square disk on a peg, pouring beads from one cup to another, and stirring in a bundt-cake pan.



Procedure—Following informed consent, participants were provided with information on the goals of the study and were invited to ask any questions they may have. The participants first put on a velcro motion-capture glove, stood in a fixed location behind the robot, and waited in a comfortable initial pose with their palm level to the floor and fingers facing forward. The standing spot was selected to provide a sufficient vantage point for all tasks and to ensure that the participants would be out of the robot's range of motion at all times for safety. The experimenter then guided the participants through a practice phase on how to control the Hubo robot. The system was initialized by the experimenter, whereby a text-to-speech engine counted down from five to signal when the participant would have control. Once the system counted down, the participant could move his/her arm and hand in free-space to practice using the mimicry-control system by picking up and moving around an empty water bottle for up to four minutes.

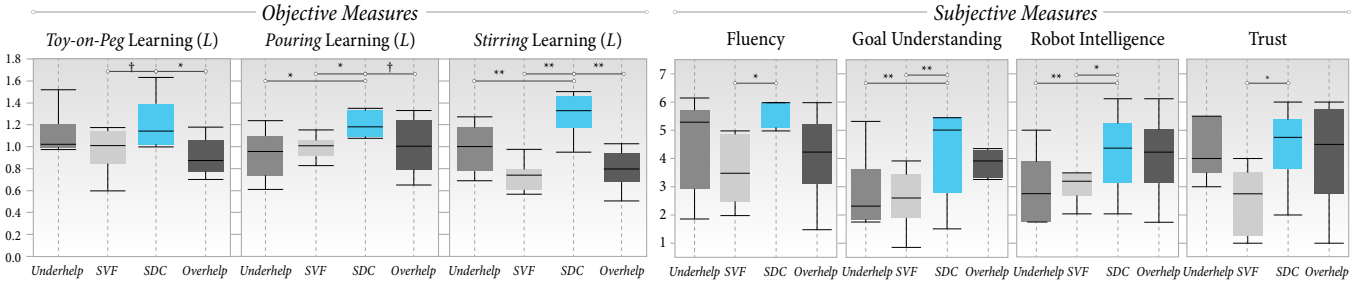
After the participants felt sufficiently comfortable using the mimicry-control system, they took a short break while the experimenter set up the first of three tasks. The task order was randomized for each participant. Once the task was understood, the participant performed the task six times, with a short break and reset of the robot and task objects between each trial. They were given a maximum time limit of two minutes for each of the seven training trials. During the first and last trials in all conditions and tasks, the participant controlled the robot using direct mimicry control in order to measure the participant's motor-control skill before and after training. The four trials between these trials were *training trials*, where the training strategy depended on to which training condition, outlined below, the participant was assigned. The participants were not told that the first and last trials were evaluations or that their overall goal was to get better at the tasks over time; they were instructed to do their best at the task on each individual trial. In all conditions, the participants were told that the robot *may* try to assist during the tasks and the control method may change between trials, but they were not told when or whether or not the robot would assist. This procedure was repeated for the remaining two tasks. Each task was preceded by a short break. After finishing the three tasks, the participants completed a questionnaire and were then debriefed on the details of the study.

Tasks—Participants performed three tasks—*toy-on-peg*, *pouring*, and *stirring*—presented in a randomized order (Figure 4). For each task, the robot had a pre-set initial configuration, and the task objects were placed in known positions. In the *toy-on-peg* task, the participants used the robot to grasp a small square disk secured on a peg, slide the disk up off of its peg, then thread the block onto another nearby peg. The idealized curves for the *toy-on-peg* task were straight lines between task-segment goal points. The idealized rotation curve was static at the origin, which still allowed flexing and bending away from the origin curve in rotation space. The *pouring* task required participants to control the robot to pick up a plastic cup, pour eight beads into another cup, then place the cup back in its original position on an upside-down cup. The idealized curves for this task were straight lines in operational space between the task-segment goal points. For the pouring motion, the rotation curve was specified to allow pouring at the correct orientation when the other cup was approached. In the *stirring* task, the participants controlled the robot to stir five times around the central point of a bundt-cake pan. The participants were told to stir as quickly as they could without hitting the edges of the pan. The idealized curves for this task consisted of a single, circular curve. The rotation curve for this task was static at the origin as in the *toy-on-peg* task.

Training Approaches—In our study, participants were randomly assigned to one of four training approaches: *Shared Dynamic Curves*, *Underhelping*, *Overhelping*, and *Scaffolded Virtual Fixtures*.

The *Shared Dynamic Curves* approach used the shared-control methods we have described in this paper. Through the four training trials, the scaffolding value defined in §4.4 was set to 0.1, 0.4, 0.7, and 1.0, respectively, to cover the full spectrum of robot assistance. In the *Underhelping* condition, participants used the mimicry-control method as described by Rakita et al. [21], which served as the direct-control baseline. In the *Overhelping* condition, the assistance was turned all the way up for each of the four training trials by setting the SDC scaffolding value defined in §4.4 to 1.0. This training method represented shared-control paradigms that continuously provide user assistance and do not aim to support motor task learning. Because both of these conditions lack scaffolding, these comparisons were included to assess whether or not scaffolding provides a significant benefit in motor-task training. In the *Scaffolded Virtual Fixtures (SVF)* condition, we applied the same concept of scaffolding present in SDC but implemented using a virtual-fixtures approach following the geometric formulation described by Marayong et al. [16]. This condition used values of 0.1, 0.4, 0.7, and 1.0 to mix in the user's velocity inputs (k_u in Figure 2-b) for the four respective training trials and used a static value of $v_b = 0.2$ as the vector that draws the control point back to the idealized curve. These values were selected through pilot testing to elicit similar levels of assistance as the SDC condition. The SVF implementation used the same pre-defined idealized curves utilized

Figure 5: Tukey boxplots of data from objective measures (top) for each training method across tasks and subjective measures (bottom) across training methods. †, *, ** denote $p < .10$, $p < .05$ and $p < .01$, respectively.



by SDC. We included this comparison to understand whether or not the technical solutions offered by the SDC method improve the effectiveness of scaffolding in motor-task training by overcoming potential limitations of SVF presented in §3.2.

5.4 Measures & Analyses

To assess participants' motor-control learning on the three study tasks, we measured direct mimicry-control performance on the tasks before and after training using a compound objective measure that captured how the users' skill changed over time. This measure, L, has the general form $L = (1 + p_a)/(1 + p_b)$, where p_a and p_b are the participant's performance after and before training, respectively. A high value for this metric signifies an improvement on task skill, and a low value signifies diminishing performance. To calculate p_a and p_b, we use a compound measure that incorporates task proficiency and completion time in order to represent motor learning to have occurred on a task if the user both performed the task more proficiently and faster, not just one or the other. For example, in the *stirring* task, a faster time in the post-training measurement should not signify task improvement if the user hit the edge of the bowl with the spoon twice as often.

For the *toy-on-peg* and *pouring* tasks, the compound measure takes the form $p_{\{a,b\}} = (\frac{t_{max} - t_{\{a,b\}}}{t_{max}}) * (\frac{s_{\{a,b\}}}{s_{max}})$. Here, t_{max} is the maximum allowable time per training trial; $t_{\{a,b\}}$ signifies the participant's task completion time after (a) or before (b) training; s_{max} is the total number of task segments for a given task; and $s_{\{a,b\}}$ is how many task segments the participant completed after (a) or before (b) training. The total time t_{max} was two minutes for all tasks. The number of task segments, s_{max}, for the *toy-on-peg* task was three, with segments *grasp block*, *lift block off of peg*, and *drop block on other peg*. The number of task segments, s_{max}, for the *pouring* task was four, including *grasp cup*, *approach other cup*, *pour at least five beads into other cup*, and *balance cup on starting cup*.

In the *stirring* task, p_a and p_b have a similar form to *toy-on-peg* and *pouring*, but the task-segment-based measurement of task proficiency is replaced by the number of edge hits while stirring: $p_{\{a,b\}} = (\frac{t_{max} - t_{\{a,b\}}}{t_{max}}) * (\frac{e_{max} - e_{\{a,b\}}}{e_{max}})$. Here, e_{max} is an upper bound on counted edge hits, and $e_{\{a,b\}}$ is the number of times the participant hit the edge after (a) or before (b) training. For all three tasks, all terms in $p_{\{a,b\}}$ are normalized such that $p_{\{a,b\}} \in [0, 1]$. Participants were told that time and task proficiency were equally important, thus the two terms are equally weighted in the metric.

To measure participants' perceptions of their robot teacher under different training approaches, we administered a questionnaire including scales to measure *fluency*, *robot intelligence*, *trust*, and *goal understanding*. Each scale was measured on a seven-point rating scale (1 = "Strongly disagree;" 7 = "Strongly agree"). Fluency was measured using items "The robot and I worked fluently together as a team" and "The robot contributed to the effectiveness of our team" (Cronbach's α = 0.78), which were adapted for a control setting from the HRC fluency scale used by Hoffman [7]. Robot intelligence was measured using items "I felt the robot was intelligent," "The robot was able to independently make decisions through the tasks," and "The robot had an understanding of the task" (Cronbach's α = 0.93). Trust was measured using items "I trusted the robot to do the right thing at the right time" and "The robot was trustworthy" (Cronbach's α = 0.89). Lastly, goal understanding was measured using items "The robot perceives accurately what my goals are," The robot does not understand what I am trying to accomplish" (inverted), and "The robot and I are working towards mutually agreed upon goals" (Cronbach's α = 0.81). Both trust and goal understanding are scales developed by Hoffman [7].

We analyzed data from all measures using one-way analyses of variance (ANOVA). Data from the objective learning measures were analyzed for each task independently, and data from subjective measures were analyzed between conditions. All pairwise comparisons between SDC and the three alternative training methods used Bonferroni Correction by multiplying the p-value generated from Student's t-test by three. To maximize readability of the reporting in §5.6, all comparisons are shown in Figure 5.

5.5 Participants

We recruited 32 volunteers (19 male, 13 female) from the campus of the Naval Research Laboratory in Washington, D.C. Participant ages ranged 20–60 (M = 31.25, SD = 8.99). Participants reported a relatively high familiarity with robots (M = 4.53, SD = 1.70, measured on a seven-point scale). Nine participants had participated in a prior robotics study.

5.6 Results

Our analysis showed that the extent to which our hypotheses were supported was task dependent. In the *stirring* task, our method afforded significant learning effects over all other approaches, fully supporting H1. Data from the *pouring* task provided partial support for H1, as our method offered significant improvements over

Underhelping and *Scaffolded Virtual Fixtures* but only a marginal improvement over *Overhelping*. Lastly, in the *toy-on-peg* task, H1 found partial support, as we found a significant learning improvement over *Overhelping*, a marginal improvement over *Scaffolded Virtual Fixtures*, and no significant differences with *Underhelping*.

Data from our subjective measures provided partial support for H2. Participants found the robot marginally more *fluent* when they used our method over *Scaffolded Virtual Fixtures*, and no significant differences were found between our method and the other two conditions. Participants rated the *goal understanding* of the robot significantly higher when they used our method over *Underhelping*, and no significant differences was observed between our method and the other two comparison conditions. Data from the *robot-intelligence* measure provided more support for H2; as the robot was perceived to be significantly more intelligent when participants used our method over *Underhelping* and *Scaffolded Virtual Fixtures*, although no difference was observed between our method and *Overhelping*. Lastly, participants reported significantly higher trust when they used our method over *Scaffolded Virtual Fixtures*, as predicted by H2, but there were no differences between our method and the other comparison conditions.

5.7 Discussion

Our results provide substantial support for the central premise that SDC affords effective motor task training but indicate that the extent of these benefits are task dependent. We see the most pronounced results in the *stirring* task, followed by the *pouring* task, and the differences were the least significant in the *toy-on-peg* task. As a potential explanation of these differences, we observed that the tasks went from most difficult to least difficult for participants upon first try in the order above. For instance, 72% of participants could do the full *toy-on-peg* task using direct mimicry control on the first try, compared to 34% of participants who could do the full *pouring* task and 0% of participants who could do the *stirring* task without hitting the edge on the first try. These differences suggest that the more complex and difficult the motor task was for the user on first try, the stronger learning was using an effective scaffolding method such as SDC. We speculate that SDC did not show stronger learning effects in the *toy-on-peg* task, because this task was not in most participants' zone of proximal development. Because the goal of scaffolding is to help traverse a user's zone of proximal development by changing a task that could previously only be done with assistance into a task that can be completed independently, the benefits of the assistance are diminished if the task can already be done independently by many users.

Our subjective results also partially support our second hypothesis and help shed further light on why different learning effects may be observed between different training approaches. We found that participants who used the SDC method found the robot to be more trustworthy than those using the SVF approach did. Trust is an important consideration for the effectiveness of shared control, which is particularly noted by Nikolaidis et al. [19]. We believe that this lack of trust can be attributed to the drawbacks noted in §3.2, such as skating past the goal or discontinuous motion. For instance, guiding the end effector beyond the rotation goal in the pouring task resulted in spilled beads on the floor. Users likely lost trust in the robot in this case, jeopardizing the fragile teacher-pupil collaboration between the robot and participant. Our method also led participants to perceive the robot as being more intelligent than scaffolded virtual fixtures and underhelping did, because the robot clearly indicated its understanding of user goals by effectively blending the user's inputs with the idealized curves.

6 GENERAL DISCUSSION

In this paper, we presented a method that improves users' motor ability in shared-control telemanipulation. Our method, *Shared Dynamic Curves*, utilizes a novel dynamical-system algorithm that draws on the learning-theory concept of scaffolding. This approach involves the robot initially aggressively assisting the user, then gradually relinquishing control to the user as user skill in the task improves. Our approach can be utilized in job-training scenarios, such as training caretakers for tele-nursing or remote home health care. For instance, a nurse could first train to remotely unscrew caps off of pill bottles using shared control where object locations are preset and known, then move on to direct control in unstructured real-world settings once task proficiency is reached.

Our user study established the effectiveness of our method in training participants in certain motor tasks using a direct-control interface. Four high-level takeaways result from our work: (1) a shared-control telemanipulation system can allow a robot to effectively teach its user how to improve at a motor task; (2) the robot can achieve this goal by gradually handing control over to the user across a series of training trials based on the learning-theory concept of scaffolding; (3) our novel shared-control method, *Shared Dynamic Curves*, provides better support for scaffolding in a shared-control setting than alternative approaches such as virtual fixtures, increasing users' motor task learning; and (4) the benefits of scaffolding for motor task learning are more pronounced in tasks that are initially more difficult for the user.

Several aspects of the current work suggest future avenues of research. While we aimed to develop a motor task training method that can help stroke or traumatic-brain-injury patients to regain motor abilities in rehabilitation, we note that the tasks in our study followed a job-training scenario. Although we showed the effectiveness of our method in improving learning, we have not yet shown its efficacy as a tool to regain generalized motor-control ability. In our future work, we plan to test whether or not our method generalizes to the broader scope of motor task training.

Finally, further research is necessary to establish whether or not the learning gains shown in our study are retained over long periods of time and whether or not they can be obtained in real-world job training and rehabilitation scenarios. We plan to further develop our method to support such scenarios and test its long-term effectiveness in our future work.

7 ACKNOWLEDGMENTS

Funding for this work was provided by the Office of Naval Research through an award to Laura M. Hiatt. The authors would like to thank Tony Harrison, Magdalena Bugajska, and Greg Trafton for their valuable help and feedback. The views and conclusions contained in this paper do not represent the official policies of the U.S. Navy.

REFERENCES

[1] Daniel Aarno, Staffan Ekvall, and Danica Kragic. Adaptive virtual fixtures for machine-assisted teleoperation tasks. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1139–1144. IEEE, 2005.

[2] Seth Chaiklin. The zone of proximal development in vygotsky's analysis of learning and instruction. *Vygotsky's educational theory in cultural context*, 1: 39–64, 2003.

[3] Jacob W Crandall and Michael A Goodrich. Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1290–1295. IEEE, 2002.

[4] Anca D Dragan and Siddhartha S Srinivasa. A policy-blending formalism for shared control. *The International Journal of Robotics Research*, 32(7):790–805, 2013.

[5] David Feygin, Madeleine Keehner, and R Tendick. Haptic guidance: Experimental evaluation of a haptic training method for a perceptual motor skill. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on*, pages 40–47. IEEE, 2002.

[6] Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688–1703, 1985.

[7] Guy Hoffman. Evaluating fluency in human-robot collaboration. In *International conference on human-robot interaction (HRI), workshop on human robot collaboration*, volume 381, pages 1–8, 2013.

[8] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization for teleoperation and teaming. *arXiv preprint arXiv:1706.00155*, 2017.

[9] Hyun K Kim, J Biggs, W Schloerb, M Carmena, Mikhail A Lebedev, Miguel AL Nicolelis, and Mandayam A Srinivasan. Continuous shared control for stabilizing reaching and grasping with brain-machine interfaces. *IEEE Transactions on Biomedical Engineering*, 53(6):1164–1173, 2006.

[10] David Kortenkamp, Debra Keirn-Schreckenghost, and R Peter Bonasso. Adjustable control autonomy for manned space flight. In *Aerospace Conference Proceedings, 2000 IEEE*, volume 7, pages 629–640. IEEE, 2000.

[11] Gert Kwakkel, Boudewijn J Kollen, and Hermano I Krebs. Effects of robot-assisted therapy on upper limb recovery after stroke: a systematic review. *Neurorehabilitation and neural repair*, 22(2):111–121, 2008.

[12] Peter Langhorne, Julie Bernhardt, and Gert Kwakkel. Stroke rehabilitation. *The Lancet*, 377(9778):1693–1702, 2011.

[13] Dale A Lawrence. Stability and transparency in bilateral teleoperation. *IEEE Transactions on Robotics and Automation*, 9(5):624–637, 1993.

[14] Jehee Lee. Representing rotations and orientations in geometric computing. *IEEE Computer Graphics and Applications*, 28(2):75–83, 2008.

[15] Ming Li and Allison M Okamura. Recognition of operator motions for real-time assistance using virtual fixtures. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings. 11th Symposium on*, pages 125–131. IEEE, 2003.

[16] Panadda Marayong, Ming Li, Allison M Okamura, and Gregory D Hager. Spatial motion constraints: Theory and demonstrations for robot guidance using virtual fixtures. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1954–1959. IEEE, 2003.

[17] Katharina Muelling, Arun Venkatraman, Jean-Sebastien Valois, John Downey, Jeffrey Weiss, Shervin Javdani, Martial Hebert, Andrew B Schwartz, Jennifer L Collinger, and J Andrew Bagnell. Autonomy infused teleoperation with application to bci manipulation. *arXiv preprint arXiv:1503.05451*, 2015.

[18] Günter Niemeyer, Carsten Preusche, and Gerd Hirzinger. Telerobotics. In *Springer Handbook of Robotics*, pages 741–757. Springer, 2008.

[19] Stefanos Nikolaidis, Yu Xiang Zhu, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in shared autonomy. *arXiv preprint arXiv:1701.07851*, 2017.

[20] Shahin S Nudehi, Ranjan Mukherjee, and Moji Ghodoussi. A shared-control approach to haptic interface design for minimally invasive telesurgical training. *IEEE Transactions on Control Systems Technology*, 13(4):588–592, 2005.

[21] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. A motion retargeting method for effective mimicry-based teleoperation of robot arms. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 361–370. ACM, 2017.

[22] Louis B Rosenberg. Virtual fixtures: Perceptual tools for telerobotic manipulation. In *Virtual Reality Annual International Symposium, 1993 IEEE*, pages 76–82. IEEE, 1993.

[23] R Keith Sawyer. *The Cambridge handbook of the learning sciences*. Cambridge University Press, 2005.

[24] Joe Warren, Scott Schaefer, Anil N Hirani, and Mathieu Desbrun. Barycentric coordinates for convex sets. *Advances in computational mathematics*, 27(3):319–338, 2007.

[25] Erkang You and Kris Hauser. Assisted teleoperation strategies for aggressively controlling a robot arm with 2d input. In *Robotics: science and systems*, volume 7, page 354, 2012.