Inferring and Assisting with Constraints in Shared Autonomy

Negar Mehr, Roberto Horowitz, Anca D. Dragan

Abstract— **Our goal is to enable robots to better assist people with motor impairments in day-to-day tasks. Currently, such robots are teleoperated, which is tedious. It requires carefully maneuvering the robot by providing input through some interface. This is further complicated because most tasks are filled with constraints, e.g. on how much the end effector can tilt before the glass that the robot is carrying spills. Satisfying these constraints can be difficult or even impossible with the latency, bandwidth, and resolution of the input interface. We seek to make operating these robots more efficient and reduce cognitive load on the operator. Given that manipulation research is not advanced enough to make these robots autonomous in the near term, achieving this goal requires finding aspects of these tasks that are difficult for human operators to achieve, but easy to automate with current capabilities. We propose constraints are the key: maintaining task constraints is the most difficult part of the task for operators, yet it is easy to do autonomously. We introduce a method for inferring constraints from operator input, along with a confidence-based way of assisting the user in maintaining them, and evaluate in a user study.**

I. Introduction

Decades ago, industrial robots were developed to automate dirty, dull, and dangerous tasks, and they purposely eschewed human involvement for reasons of speed and safety. In stark contrast, a new generation of medical and health care robots is designed for direct interaction with human users in environments such as the surgical theater, the rehabilitation center, and the family room. Teleoperated robots such as the da Vinci Surgical System are used to perform minimally invasive surgery around the world, resulting in shorter recovery times and more reliable outcomes in select procedures [1]–[3]. Physically assistive systems are being developed for in-home use to help limited mobility users with Activities of Daily Living (ADLs) [4]–[7].

These robots have the potential to help doctors and patients alike with performing tasks that are difficult for them to accomplish with their own body: robots have access to quantitative data about tasks and environments, and can move accurately without tiring. Despite this, integrating them with human users still presents significant challenges in performance and user acceptance, for two reasons. On the one hand, teleoperating and controlling these robots is tedious and does not take advantage of many of the potential advantages of artificial intelligence and robot mechanisms. On the other hand, current manipulation research is not advanced enough to make these robots autonomous in medical and health care environments in the near term.

Our insight is that there are important aspects of manipulation tasks that are difficult for human operators to achieve, but relatively easy to automate with current capabilities.

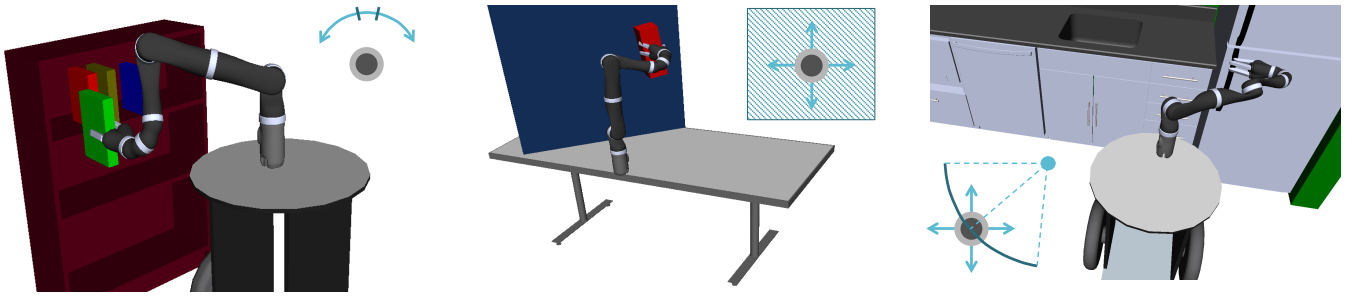
Rather than attempting full automation, we propose to automate these aspects. We focus on task *constraints* as the key, because maintaining task constraints is the most difficult part of many tasks for human operators, yet autonomous systems can easily keep track of and satisfy multiple constraints. Figure 4 shows examples of constraints encountered in ADLs, including orientation and position constraints: in order to place a book in the book shelf, its orientation needs to be aligned with the other books; to wipe a board, the end–effector is constrained to move along the plane of the board; or the end–effector has to traverse an arc when opening the door of a refrigerator.

Two challenges make assisting human operators with maintaining tasks constraints difficult: 1) inferring which constraint the operator is trying to enforce, and 2) deciding how to provide assistance once a constraint is inferred. Our contributions are in line with these challenges:

Constraint Inference: Typically, constraint inference is done from multiple demonstrations of the same task [8]–[11]. In shared autonomy, however, the robot needs to infer the constraint from a *single ongoing execution* of the task. We introduce a constraint inference algorithm, capable of online inference of task constraints from the ongoing stream of user's control inputs and trajectory of the robot in the task space. Our approach is based on the Mowing Window KPCA [12], a variant of Kernel Principle Component Analysis (KPCA) [13]. This enables the robot to handle *non-linear* constraints *online*, inferring the constraint solely based on the most recent control inputs in order to make the computation real-time.

Moving Window KPCA identifies a constraint in feature space. To enforce the constraint, the robot has to first map the user input into feature space, and then project it onto the constraint. This gives the robot a direction in *feature* space, but does not identify the constraint in *task* space. We contribute a preimage learning technique that learns to approximately project a feature space direction back in the robot's task space, which the robot can then follow (e.g. via Jacobian-based or IK-based controllers).

Assisting with (Changing) Constraints: Given a predicted constraint, the robot has to decide how and how much to assist the user. One option is to enforce the



(a) Orientation Constraint (b) Position Constraint - Plane (c) Position Constraint - Distance

Fig. 1: Example constraints for ADL-assistive robotics with a Kinova arm that will be mounted to a wheelchair.

predicted constraint, but this can be catastrophic if the robot has the wrong prediction. Building on [14], we propose to mediate assistance by the robot's confidence in the constraint. We introduce a confidence measure capable of dealing with the fact that constraints may be changing over time, or that the user might be going from no constraint to a constraint and vice-versa.

User Study Evaluation: We conduct two user studies with novice users in which we measure the effectiveness of our algorithm in assisting users with maintaining task constraints. When dealing with a single constraint, our results support that users prefer to be assisted by the robot, and their objective measure of performance improves significantly by means of assistance. When dealing with switching constraints, our measure of confidence enables the robot to stop assisting and make it easier for the user to perform the switch, improving subjective and objective measures.

II. Related Work

Shared Autonomy. The very first instance of shared autonomy (or shared control) was in manipulators utilized for transporting radioactive material, where the low accuracy of user control input can lead to violating safety criteria [15]. Since then, various methods have been introduced [16]–[21].

[14] characterized the problem of shared control in tasks where there is continuous user input as arbitrating between the user input and a robot predicted policy, showing how many algorithms instantiate this characterization by using different predictors and different arbitration functions. This is the paradigm we adopt as well. However, instead of predicting the operator's desired goal and assuming that the robot can achieve it as in [14], we predict desired *constraints*. This requires more limited robot capability, yet still captures a difficult aspect of the task for the operator.

Virtual Fixtures. Virtual fixtures [22] are the most related type of shared autonomy to the topic of this work. Virtual fixtures assist the operator with maintaining constraints by constraining or guiding either the user's input or the robot itself [23], [24]. Implementations in abstract and surgical environments have demonstrated the effectiveness of static virtual fixtures, as well as the role of haptic feedback in two-degree-of-freedom guidance [25]–[27].

Thus far, the assumption has been that the robot *knows* the constraint (fixture) to maintain. In this work, we enable the robot to *infer* it from operator data and control inputs. This is not only an inference challenge, but also an assistance challenge because the robot is no longer certain of what fixture or constraint the user wants.

Constraint Inference. Although inferring constraints has not been actively studied in teleoperation, it has been a focus in Learning from Demonstration [28]. There, constraints are identified as the invariants among a set of demonstrated trajectories for the same task [8]–[11].

Unlike Learning from Demonstration, our robot does not get access to multiple training trajectories for the same task. Instead, in teleoperation, the robot needs to infer the constraint being maintained along a trajectory online, as the trajectory is being observed.

III. Problem Statement

In this section, we formulate the problems of constraint inference and assistance.

A. Online Constraint Inference

Let $s \in \mathcal{S}$ denote the state, here the end-effector task space in SE(3) and $u_t \in \mathcal{U}$ be the control input that the user provides (via some interface, e.g. a joystick) at time t. We represent a constraint that a user might attempt to satisfy, e.g. keeping a bottle of water upright, as the zero set of some function h of the state s. The set of states satisfying a constraint c denoted by $S_c \in \mathcal{S}$ is:

$$S_c = \{s | h(s) = 0\}, \quad h : \mathcal{S} \rightarrow \mathbb{R}^+ \tag{1}$$

Given the operator's control input from the starting time to the current time step t, $u_0, \cdots, u_t$, and the robot trajectory so far, $s_0, \cdots, s_t$, the constraint inference problem requires finding the function h_t that characterizes the constraint the operator currently wants to maintain.

B. Assistance with Maintaining Constraints

Once the robot has an inference of the constraint, the question is how to deploy this prediction in assisting the user. Particularly, given the current state s_t, the user input u_t, and the current prediction h_t, the robot needs to decide whether and how to modify u_t to assist the user.

IV. Inferring and Assisting with Constraints

A. Online Constraint Inference

Given the states, $L_t = \{s_0, \cdots, s_t\}$, that noisily satisfy a constraint h_t, along with the user input vector $\{u_0, \cdots, u_t\}$ the goal is to infer h_t. Because the zero set of h_t is a manifold in $\mathcal{S}$, manifold learning is a natural choice for identifying the constraint.

Manifold learning algorithms attempt to find a lower dimensional representation of given points such that a certain proximity criteria is met in a lower dimensional subset of data points. There exists a large number of manifold learning algorithms [29]–[33] that could possibly be utilized in inferring h_t. The distinction between the existing manifold learning algorithms arises from the type of proximity criteria that they try to optimize, or the class of functions they can learn for h_t.

Constraint inference raises some requirements for manifold learning that narrow down which algorithms are applicable. First, manifolds might be *non-linear*. Second, manifolds need to be learned *online*, in real time, as the operator's input data is coming in.

Kernel PCA The requirements above point to *Kernel Principle Component Analysis* [13] (KPCA) as the basis of our work. In this method, principle components can be found in a non–linear feature space defined by the choice of a kernel function. KPCA maps the data points to a non–linear, possibly infinite dimensional feature space F, through a non–linear feature mapping ϕ:

$$s \mapsto \phi(s) \in F,$$

and computes principle components of the mapped data points, $\{\phi(s_0), \cdots, \phi(s_t)\}$, in the obtained feature space.

Nominally, this would be achieved by computing the eigenvectors of the covariance matrix C of mapped data points in F:

$$C_t = \frac{1}{t}\sum_{i=1}^{t} \phi(s_i)\phi(s_i)^T \tag{2}$$

The main idea behind KPCA is to compute the principal components without ever mapping the data to feature space, or even *explicitly* defining ϕ. Instead, KPCA operates by introducing a kernel function k and *implicitly* defining ϕ such that $k(s_i, s_j) = \langle \phi(s_i), \phi(s_j) \rangle$. It constructs a $t \times t$ kernel matrix K via $K_{ij} = k(s_i, s_j)$. Each principle component, v_l, can be written as a linear combination of $\phi(s_i)$'s:

$$v_l = \sum_{i=1}^{t} \alpha_i^l \phi(s_i). \tag{3}$$

The vector of coefficients α_i^l, denoted by $\boldsymbol{\alpha}^l$, can be computed solely from K by solving:

$$t\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha}, \tag{4}$$

where λ is an eigenvalue of K, and $\boldsymbol{\alpha}$ is its corresponding eigenvector.

Note that in the above, for the sake of simplicity, we have assumed that the data is centered in the feature space, i.e. $\sum_{i=1}^{t} \phi(s_i) = 0$. In general, this does not need to hold. In that case, we need the eigendecomposition of the Gram matrix G instead of K:

$$G = K - \frac{1}{t}KE_t - \frac{1}{t}E_tK + \frac{1}{t^2}E_tKE_t, \tag{5}$$

where E_t is the $t \times t$ matrix with all entries equal to 1. See [12] for more details.

Note that the output of KPCA is the set of vectors $\boldsymbol{\alpha}$. As a result, the projection of a state s on the l^{th} kernel principle component of data points can be computed from $\boldsymbol{\alpha}_l$ using the kernel function, without explicitly computing $\phi(s)$:

$$\langle v_l, \phi(s) \rangle = \sum_{i=1}^{t} \alpha_i^l \langle \phi(s_i), \phi(s) \rangle \tag{6}$$

$$= \sum_{i=1}^{t} \alpha_i^l k(s_i, s), \tag{7}$$

where α_i^l is the i^{th} entry in the vector of αs corresponding to the l^{th} kernel principle component.

Projecting s on h_t in *feature* space means projecting $\phi(s)$ onto the first D principle components according to the largest D eigenvalues:

$$Ps = \sum_{l=1}^{D} \frac{\langle v_l, \phi(s) \rangle}{\langle v_l, v_l \rangle} v_l \tag{8}$$

We discuss raising Ps back into task space in "Enforcing an Inferred Constraint" below.

Online KPCA Running KPCA over the entire set of user–provided states and control inputs is computationally prohibitive. We make KPCA online by using a finite time horizon: a moving window where the newest user input point replaces the oldest one, and *we reuse computation* from the previous step whenever the window moves.

We perform KPCA for a rolling constant–length horizon of M data points, $L_t = \{s_{t-M+1}, \cdots, s_t\}$. This implies that at time $t+1$, a new data point, s_{t+1} is received and the very first element of the horizon of data, s_{t-M+1}, is removed from the history of data points. This updates the matrix K.

Running KPCA over a time horizon of M still requires the eigendecomposition of K at each step, which

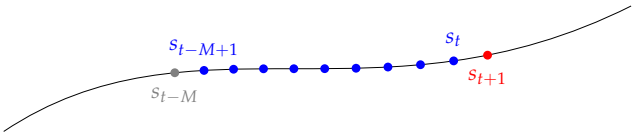


Fig. 2: We do online constraint inference by leveraging previous computation (for $\{s_{t-M} \cdots s_t\}$) to infer the constraint based on the current horizon ($\{s_{t-M+1} \cdots s_{t+1}\}$).

is in $\mathcal{O}(M^3)$. To overcome this computational hindrance of KPCA, we need to reuse the computation performed in the previous time steps. Liu et al. [12] show that updating the eigendecomposition of the Gram matrix

$$G = K - \frac{1}{M}KE_M - \frac{1}{M}E_MK + \frac{1}{M^2}E_MKE_M, \quad (9)$$

at every step can be done in $\mathcal{O}(M^2)$. Here, $E_M = 1_M \times 1_M^T$ and 1_M is the M dimensional vector of 1's.

Enforcing an Inferred Constraint KPCA learns the constraint as a manifold in its Reproducing Kernel Hilbert Space F. The output of KPCA is the set of vectors $\boldsymbol{\alpha}^j$, which enable projecting the user inputed next state s_t onto each principle component, and in turn that enables computing the projection of s_t onto h_t in F, denoted by Ps_t, via (8). To enforce the constraint, the robot needs to compute the state corresponding to Ps_t, i.e. the *preimage* of the projection.

In general, finding preimages is computationally expensive. The approach we take for finding preimages is an approximation based on [34]: we learn a function $\Gamma : F \to \mathbb{R}^d$ such that $\Gamma(\phi(s)) \simeq s$ locally.

Our insight for learning a Γ is to use the robot's trajectory so far, along with the projections Ps on the currently learned manifold, as training data: $\mathcal{D} = \{(s_i, Ps_i) | i = \{t-M+1, \cdots, t-1\}\}$. A few aspects combine to makes this work well: 1) Γ only needs to work well locally, in the space of interest, i.e. along the robot's trajectory, where the training data is; 2) the states visited so far, s_i, like approximately on the manifold: even though $\phi(s_i) \neq Ps_i$ in general, this is approximately true along the trajectory so far (in the current time horizon); and 3) we use a simple function class for Γ to prevent overfitting to the noise in s_i, and construct a function that maps points in feature space closer to the actual manifold in task space.

We thus use a linear model for each output dimension of Γ:

$$\Gamma_j(\psi) = w_j^T \psi. \quad j = 1, \cdots, d \quad (10)$$

for $\psi \in F$. We learn the weights w_j by solving the following optimization problem:

$$w_j = \underset{w_j}{\operatorname{argmin}} \sum_{i=t-M+1}^{t-1} \mathcal{L}\left(s_i, w_j^T P\phi(s_i)\right) + \lambda\Omega(\Gamma), \quad (11)$$

where $\mathcal{L}$ is loss function penalizing distance in task space, Ω is a a regularizer, and λ is a pre–selected non–negative weight.

Applying the learned function Γ to the projected state s_t in feature space, Ps_t, yields the projection of the user's desired next state in task space onto the constraint:

$$\text{projection of } s_t\text{: } \Gamma(Ps_t) \quad (12)$$

B. Assistance

Assistance for One Constraint If the robot were assisting with a known constraint, then it is sufficient to go to $\Gamma(Ps_t)$. However, the robot does not actually know the constraint, and its inference might be wrong. Key to successful assistance is knowing when to (or not to) assist.

Therefore, besides inferring the constraint manifold, the robot should also compute its confidence in the inference. In the case that the user is always following the same constraint, how far away the trajectory so far is from the manifold is a good indicator of how accurate the manifold is. We thus measure the confidence in the inference as inversely proportional to the average distance between previous states and the preimages of their projects onto the manifold:

$$d_t = \sum_{i=t-M+1}^{t} |s_i - \Gamma(Ps_i)|, \quad (13)$$

The bigger d_t is, the less confident the robot must be in its prediction. Let c_t^1 denote the confidence in the inferred constraint at time t:

$$c_t^1 = e^{-\gamma_1 d_t}, \quad (14)$$

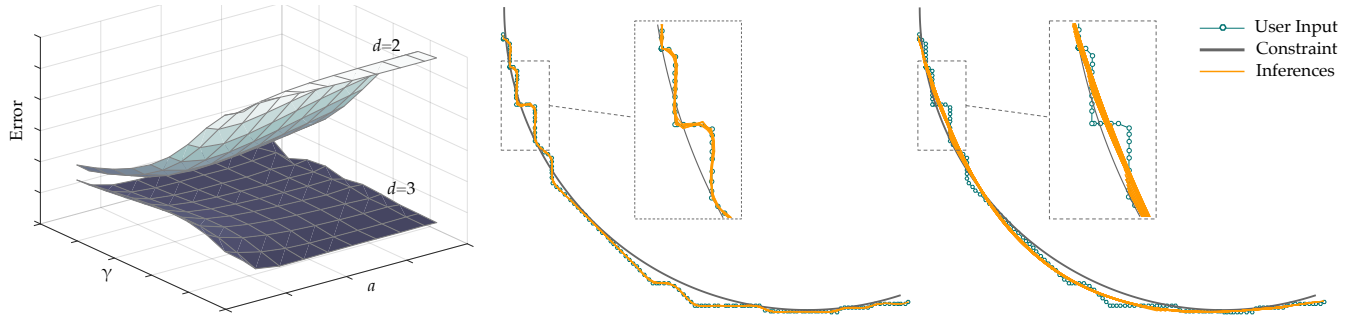
where γ_1 is a positive selected scalar. Equation (14) ensures that $c_t^1 \in [0, 1]$, and c_t^1 decreases as d_t increases.

We propose to blend between the user–provided s_t and its projection on the inferred constraint, $\Gamma(Ps_t)$, using:

$$s_t^R = (1 - c_t^1)s_t + c_t^1\Gamma(Ps_t) \quad (15)$$

with s_t^R being the blended state to which the robot moves at time t. Once s_t^R is reached, the user provides the next input u_{t+1}, leading to the inputed state s_{t+1}, and the process repeats for $L_{t+1} = s_{t-M+2}, \cdots, s_{t+1}$.

Assistance for Switching Constraints The confidence measure c^1 is designed for when there is a single constraint that the user is always trying to maintain. But executing a real task may require either maintaining a sequence of dissimilar constraints, or switching from maintaing a constraint to being unconstrained. The confidence measure c^1 does not capture such cases well: once the robot becomes confident, it starts assisting a lot and biasing the state sequence. Confidence could also be computed based solely on the user inputs; however, we found that once people realize that the robot is intelligent enough to guide them through the constraint, they stop providing informative inputs. For instance, they constantly move the joystick along a single Cartesian direction even they need the robot



(a)Polynomial Kernel Degree (b) Short Horizon (c) Long Horizon

Fig. 3: Effects of kernel parameters (a) and inference horizon (b,c) on performance. On average across our tasks, a degree 3 polynomial kernel was most robust. Short horizons lead to overfitting, with vastly different inferred constraints over time. Longer horizons are more computationally demanding but better at recovering a constraints close to the desired one. On the other hand, making the horizon too long makes switching constraints difficult, which is where our confidence measure helps.

to move along a direction different from their input, making input–based confidence impractical.

Thus, while c^1 works well in the one constraint case (as we will see in the first user study), we need to augment it to detect changing constraints. *We leverage the natural difference in operator input when following a constraint versus switching constraints*.

When the operator is controlling the robot on an ongoing constraint, once a deviation from the underlying constraints is observed, the operator provides control inputs that correct and cancel out the observed deviation. In other words, moving along a constraint is formed by a set of *deviations* and *corrections* that are on average in the direction of the constraint. On the other hand, while leaving a constraint, the states ensuing from the user's raw (un–arbitrated) inputs are constantly along a direction different from h_t. We can detect this.

This suggests that augmenting our measure of confidence requires computing the alignment between the direction suggested by the average user input, and the direction of the constraint. We denote the direction of average user input by δ_t^u, and compute it as

$$\delta_t^u = \frac{1}{M-1} \sum_{i=t-M+1}^{t-1} f(s_i, u_i) - s_i \tag{16}$$

where f is the task space dynamics model – this computes the average difference between the user inputed state and the current state.

The constraint manifold is feature space, so we approximate it via δ_t^s, the average direction of the robot's trajectory:

$$\delta_t^s = \frac{1}{M-1} \sum_{i=t-M+1}^{t-1} s_{i+1} - s_i \tag{17}$$

Let θ_t be the alignment between these two measures:

$$\theta_t = \cos^{-1}\left(\frac{\langle \delta_t^u, \delta_t^s \rangle}{\|\delta_t^u\| \|\delta_t^s\|}\right) \tag{18}$$

We compute c_t^2 by:

$$c_t^2 = e^{-\gamma_2 \theta_t}. \tag{19}$$

We measure the overall confidence in robot's inference, c_t^3, by combining the two measures:

$$c_t^3 = c_t^1 c_t^2 \tag{20}$$

$$= e^{-(\gamma_1 d_t + \gamma_2 \theta_t)} \tag{21}$$

c^3 replaces c^1 in 15.

V. Experiments

Our experiments are two-fold: we are interested in the performance of the constraint inference algorithm, as well as its impact on real user teleoperation scenarios.

A. Constraint Inference Analysis

In order to evaluate the accuracy of our inference algorithm, we collected direct teleoperation data (without assistance).

We asked participants to teleoperate the 7-DOF arm from Figure 4 through a joystick interface for moving a grasped object along a predefined path, demonstrated to the users in a simulation environment. We asked participants to traverse the path in the shortest possible time while keeping the grasped object as close as possible to the path. We designed three different scenarios: tracking a line, an arc with a large radius and low curvature as a result and finally moving the grasped object along a semicircle of low radius and high curvature.

Analysis A key element in the performance of any kernel–based algorithm is the choice of kernel made for the algorithm. In our problem in particular, the kernel function restricts the inferred constraint, h_t, to a subset of all possible constraints, i.e. it implicitly makes assumptions about real-world tasks.

We tested our method's performance across different kernel parameter choices, manipulating kernel type

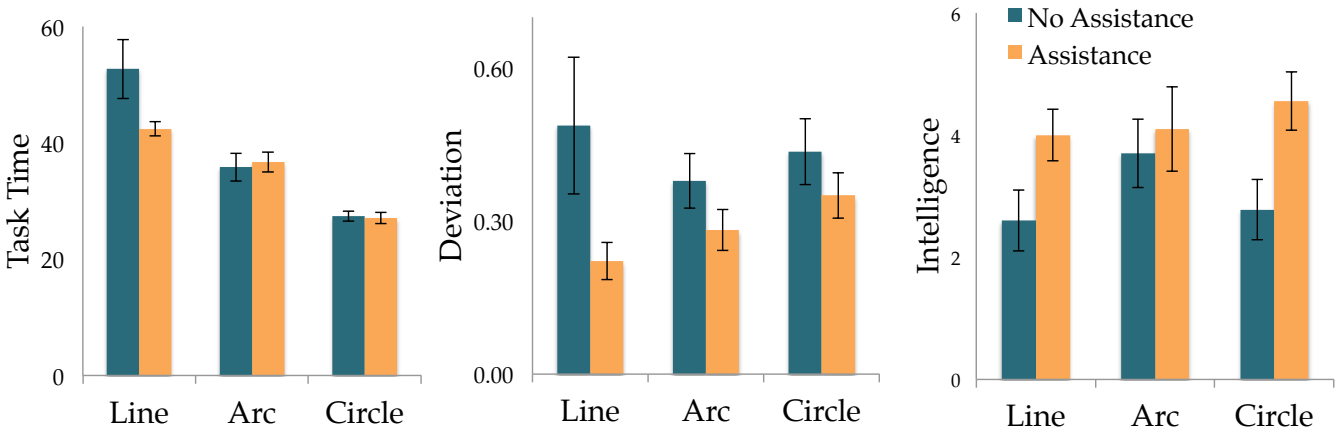


Fig. 4: Dependent measures in user study 1

(polynomial of different degrees d) and kernel parameters (coefficients γ and the free coefficient a). Across our tasks, we found the 3^{rd} order to perform the best and be more robust to parameters (Fig. 3(a)).

We also analyzed the effect of the time horizon on performance. Unsurprisingly, short time horizons produce inferred constraints that overfit to the user's noisy input (Fig. 3(b)), while longer horizons more easily recover a constraint close to the ground truth (Fig. 3(c)), canceling out the deviations in user input.

B. User Study 1: Assistance with A Single Constraint

In this first study, we assess the effectiveness of online assistance in maintaining a single constraint.

Manipulated Variables We asked participants to control an arm in the three scenarios described previously: tracking a line, an arc, and a semicircle. In each scenario, the operator controls the robot in two conditions: direct teleoperation (no assistance is provided), or the user is assisted with the inferred task constraint via our (15). We thus manipulate two factors in a factorial design:

1) Assistance: Whether or not the robot provides assistance.
2) Task Difficulty: the constraint the user needs to maintain.

This setup leads to 6 different conditions experienced by each participant. Task difficulty ranges from simple, in the line case, to hard, in the semicircle scenario. We used the confidence measure c^1 in this study.

Dependent Measures We have both objective and subjective measures:

1) Task Execution Time.
2) Average deviations from the underlying constraint. Any component of motion normal to the instantaneous direction of constraint is defined as deviation.
3) User's subjective measure of robot's intelligence for assistance vs. no–assistance in each scenario (on a 7–point Likert scale)

Hypothesis:

H1. Assistance improves the performance of the human–robot team, both objectively and subjectively.

Subject Allocation: We chose a within-subject design for 9 participants (4 females and 5 males, 27 years old on average) to enable them to compare the assistance vs. no–assistance conditions in each scenario. We counterbalanced the order of the conditions to avoid any biases resulting from ordering effects.

In order to control for the prior skill in playing with joysticks, we asked each participant to take part in a training phase to teleoperate the arm through joystick with no assistance.

Analysis: Figure 4 shows the results. We ran a factorial repeated measures ANOVA with assistance and task as factors for each of our dependent measures.

For time, we found both factors had significant effects and there was a significant interaction effect $F(2,56) = 6.1$, $p < .01$. The post-hoc analysis with Tukey HSD corrections showed that assistance improved the timing performance for the line task, $p < .01$. As the graph shows, the timing for the other two tasks were almost identical.

The results for deviations supported our hypothesis: there was a significant effect for assistance, $F(1,57) = 13.56$, $p < .01$. There was no significant effect for task and no interaction effect: assistance improved deviations across the board.

The results for rating also support our hypothesis: there was a significant effect for assistance, $F(1,57) = 17.29$, $p < .001$, and no other effects were significant. Ratings for the assistance robot were higher across the board.

Overall, the results suggest that assistance does improve subjective and objective performance. Even for

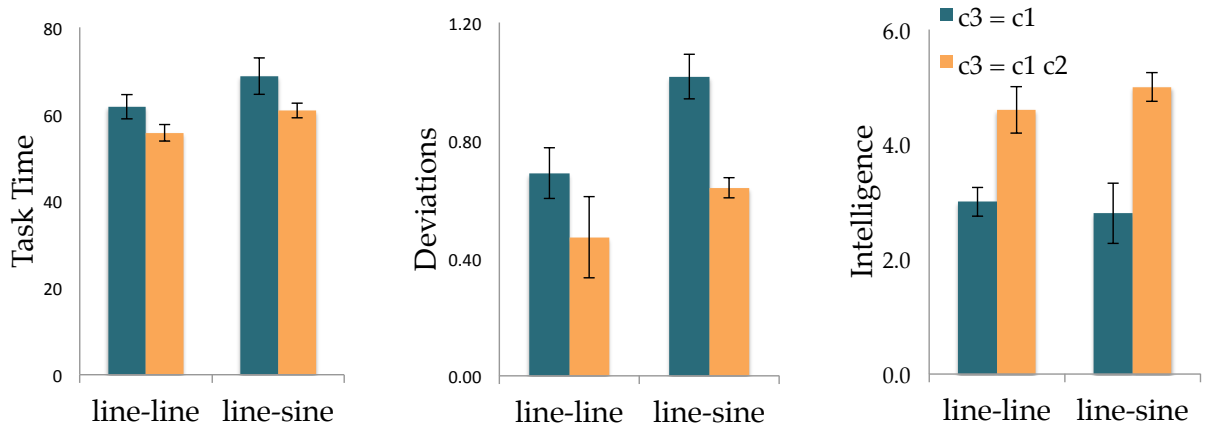


Fig. 5: Dependent measures in user study 2

simple tasks in 2D, assistance can significantly improve user's comfort. We expect that common ADL's will include 3D constraints which are much harder to satisfy through teleoperation. Furthermore, maintaining task constraints becomes much harder for people with physical impairments.

C. User Study 2: Assistance with Changing Constraints

Our first study indicated that assistance is good when dealing with a single constraint. Next, we investigated what happens when the user needs to switch from a constraint. We conducted a second study to evaluate the effect of our proposed confidence measure, c^3, on the performance of assistance.

Manipulated Variables We considered two scenarios: switching from one constraint to a a new constraint, or leaving a constraint and entering free space, thus requiring no assistance for the rest of the task execution.

1) Whether or not the confidence measure c_t^2 is taken into account ($c_t^3 = c_t^1 c_t^2$ vs. $c_t^3 = c_t^1$).
2) Whether to switch to a new constraint, or to a no–constraint scenario.

We mimicked the scenario of switching to a new constraint by designing the tracking path to be two line segments perpendicular to each other (called line–to–line scenario). We imitated requiring no–assistance from the switching point onward, by designing the path to be a line segment followed by a sequence of high curvature semicircles depicting a sinewave–like shape (line–to–sine scenario).

Dependent Measures: We used the same objective and subjective measures as before: time, deviations, and rating.

Hypothesis:

H2: Incorporating the c_t^2 measure of confidence improves the performance both subjectively and objectively.

Subject Allocation: Analogous to user study 1.

Analysis: Figure 5 shows the results. We ran a factorial repeated measures ANOVA with confidence measure and task as factors for each of our dependent measures.

For time, supporting our hypothesis, we found a significant effect for confidence measure, $F(1,27) = 6.39$, $p = .0210$: using the augmented confidence meant to handle changing constraints lead to lower task time. The task factor was also significant, with changing from one constraint to the other taking longer than changing from a constraint to no constraint. There was no significant interaction effect.

For deviations, also supporting the hypothesis, we found a significant improvement for the augmented confidence $F(1,27) = 19.01$, $p < .001$. Task again had a significant effect.

For rating, also supporting our hypothesis, we found that users rated the robot using the augmented confidence more highly, $F(1,27) = 10.47$, $p < .01$. There was no other significant effect.

Overall, the results support that our confidence metric that augments manifold-based confidence with user input direction-based confidence leads to better assistance when dealing with changing constraints.

VI. Discussion

Summary. We introduced and evaluated a method for inferring and assisting with task constraints. Constraint inference happens online, in real-time, and assistance is mediated by confidence and capable of handling changing constraints. This improved teleoperation both objectively and subjectively.

Limitations and Future Work. Albeit useful, constraints are complex and difficult to identify, and much work remains in this area: being able to restrict the space of possible constraints based on possible real-world tasks, incorporating prior knowledge about

a particular task, identifying the dimensionality of the constraint manifold automatically, handling force-based constraints, as well as testing in real-world assistive robotics scenarios, and with users with motor impairments. We are excited to tackle these challenges in future work.

VII. Acknowledgments

We thank the BDD and CITRIS centers at UC Berkeley for supporting this work, and Allison Okamura and Siddhartha Srinivasa for helpful advice.

References

[1] S. Maeso, M. Reza, J. A. Mayol, J. A. Blasco, M. Guerra, E. Andradas, and M. N. Plana, "Efficacy of the da vinci surgical system in abdominal surgery compared with that of laparoscopy: a systematic review and meta-analysis," *Annals of surgery*, vol. 252, no. 2, pp. 254–262, 2010.

[2] C. A. Peters, "Robotic surgery in pediatric urology: State of the art and future horizons," in *Pediatric Urology*, pp. 75–86, Springer, 2015.

[3] A. T. Stafford and R. M. Walsh, "Robotic surgery of the pancreas: The current state of the art," *Journal of surgical oncology*, vol. 112, no. 3, pp. 289–294, 2015.

[4] K. M. Tsui, D.-J. Kim, A. Behal, D. Kontak, and H. A. Yanco, "âĂIJi want thatâĂİ: Human-in-the-loop control of a wheelchair-mounted robotic arm," *Applied Bionics and Biomechanics*, vol. 8, no. 1, pp. 127–147, 2011.

[5] C.-H. King, T. L. Chen, Z. Fan, J. D. Glass, and C. C. Kemp, "Dusty: an assistive mobile manipulator that retrieves dropped objects for people with motor impairments," *Disability and Rehabilitation: Assistive Technology*, vol. 7, no. 2, pp. 168–179, 2012.

[6] M. J. Matarić, J. Eriksson, D. J. Feil-Seifer, and C. J. Winstein, "Socially assistive robotics for post-stroke rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 4, no. 1, p. 1, 2007.

[7] T. Röfer and A. Lankenau, "Architecture and applications of the bremen autonomous wheelchair," *Information Sciences*, vol. 126, no. 1, pp. 1–20, 2000.

[8] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pp. 255–262, ACM, 2007.

[9] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 367–372, IEEE, 2008.

[10] G. Ye and R. Alterovitz, "Demonstration-guided motion planning," in *International Symposium on Robotics Research (ISRR)*, vol. 5, 2011.

[11] L. Pais, K. Umezawa, Y. Nakamura, and A. Billard, "Learning robot skills through motion segmentation and constraints extraction," in *HRI Workshop on Collaborative Manipulation*, 2013.

[12] X. Liu, U. Kruger, T. Littler, L. Xie, and S. Wang, "Moving window kernel pca for adaptive monitoring of nonlinear processes," *Chemometrics and Intelligent Laboratory Systems*, vol. 96, no. 2, pp. 132–143, 2009.

[13] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Artificial Neural NetworksâĂŤICANN'97*, pp. 583–588, Springer, 1997.

[14] A. D. Dragan and S. S. Srinivasa, *Formalizing assistive teleoperation*. MIT Press, July, 2012.

[15] R. C. Goertz, "Manipulators used for handling radioactive materials," *Human factors in technology*, pp. 425–443, 1963.

[16] D.-J. Kim, R. Hazlett-Knudsen, H. Culver-Godfrey, G. Rucks, T. Cunningham, D. Portee, J. Bricout, Z. Wang, and A. Behal, "How autonomy impacts performance and satisfaction: Results from a study with spinal cord injured subjects using an assistive robot," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 42, no. 1, pp. 2–14, 2012.

[17] P. Marayong, A. Bettini, and A. Okamura, "Effect of virtual fixture compliance on human-machine cooperative manipulation," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2, pp. 1089–1095, IEEE, 2002.

[18] Y. Derimis and G. Hayes, "Imitations as a dual-route process featuring predictive and learning components: a biologically plausible computational model," *Imitation in animals and artifacts*, pp. 327–361.

[19] A. H. Fagg, M. Rosenstein, R. Platt, and R. A. Grupen, "Extracting user intent in mixed initiative teleoperator control," in *Proc. American Institute of Aeronautics and Astronautics Intelligent Systems Technical Conference*, 2004.

[20] D. Aarno, S. Ekvall, and D. Kragić, "Adaptive virtual fixtures for machine-assisted teleoperation tasks," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 1139–1144, IEEE, 2005.

[21] P. Aigner and B. McCarragher, "Human integration into robot control utilising potential fields," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 1, pp. 291–296, IEEE, 1997.

[22] L. B. Rosenberg, "Virtual fixtures: Perceptual tools for telerobotic manipulation," in *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, pp. 76–82, IEEE, 1993.

[23] T. Debus, J. Stoll, R. D. Howe, and P. Dupont, "Cooperative human and machine perception in teleoperated assembly," in *Experimental Robotics VII*, pp. 51–60, Springer, 2001.

[24] M. Li and A. 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*, pp. 125–131, IEEE, 2003.

[25] A. Bettini, P. Marayong, S. Lang, A. M. Okamura, and G. D. Hager, "Vision-assisted control for manipulation using virtual fixtures," *Robotics, IEEE Transactions on*, vol. 20, no. 6, pp. 953–966, 2004.

[26] H. C. Lin, K. Mills, P. Kazanzides, G. D. Hager, P. Marayong, A. M. Okamura, and R. Karam, "Portability and applicability of virtual fixtures across medical and manufacturing tasks," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 225–230, IEEE, 2006.

[27] S. B. Schorr, Z. F. Quek, W. R. Provancher, and A. M. Okamura, "Environment perception in the presence of kinesthetic or tactile guidance virtual fixtures," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pp. 287–294, ACM, 2015.

[28] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[29] I. Borg and P. Groenen, "Modern multidimensional scaling: theory and applications," *Journal of Educational Measurement*, vol. 40, no. 3, pp. 277–280, 2003.

[30] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[31] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on computers*, no. 5, pp. 401–409, 1969.

[32] N. Lawrence, "Probabilistic non-linear principal component analysis with gaussian process latent variable models," *The Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.

[33] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[34] G. H. Bakir, J. Weston, and B. Schölkopf, "Learning to find pre-images," *Advances in neural information processing systems*, vol. 16, no. 7, pp. 449–456, 2004.