Iterative Virtual Guides Programming for Human-Robot Comanipulation

Susana Sanchez Restrepo, Gennaro Raiola, Pauline Chevalier, Xavier Lamy, Daniel Sidobre

Iterative Virtual Guides Programming for Human-Robot Comanipulation

Susana Sánchez Restrepo[1,2], Gennaro Raiola[1,3,4], Pauline Chevalier[3], Xavier Lamy[1] and Daniel Sidobre[2]

***Abstract*— In human-robot comanipulation, virtual guides are an important tool used to assist the human worker by reducing physical effort and cognitive overload during tasks accomplishment. However, virtual guide's construction often requires expert knowledge and modeling of the task which restricts the usefulness of virtual guides to scenarios with unchanging constraints. To overcome these challenges and enhance the flexibility of virtual guide's programming, we present a novel approach that allows the worker to create virtual guides by demonstration through an iterative method based on kinesthetic teaching and Akima splines. Thanks to this approach, the worker is able to locally modify the guides while being assisted by them, increasing the intuitiveness and naturalness of the process. Finally, we evaluate our approach in a simulated sanding task with a collaborative robot.**

I. INTRODUCTION

The aim of human-robot comanipulation is to combine the cognitive capabilities of humans and the mechanical performances of robots to carry on exhausting tasks. Such devices are generally called *cobots*. The term *cobot* was first introduced to refer to wheeled robots using computer-controlled steering for motion guiding [Colgate et al., 1996]. Despite its specific initial meaning, the term cobot is now often used to refer to robots capable of safe physical interaction with human operators within a shared workspace. These cobotic systems must be intrinsically safe (e.g, lightweight robots with collision detection features), must reduce human physical effort [Lamy, 2011] and cognitive overload, and take advantage of the user's gesture expertise. Furthermore, cobots must be flexible enough to handle mutable process and uncertainty, while being intuitive enough to be set and programmed by non robotic-experts. In light of those assessments, the *virtual guides* approach appears as a promising solution to overcome some of the human-cobot interaction challenges by confining robot motion to the task-relevant directions [Rosenberg, 1993], [Marayong et al., 2003], though reducing physical effort and cognitive overload. Virtual guides (also known as virtual fixtures) are especially useful in contexts where human decision making is still required to perform the overall task, but where constraints on the accuracy or required forces of the motion motivate humans to perform such tasks with robot assistance [Lin et al., 2006], [Vozar et al., 2015]. Virtual guides could be functionally equivalent to fixtures in the real world. An example of a non-virtual, but real guide, is a straight edge system clamped to a wood panel to aid in making a perfectly straight cut with a circular saw. During both virtual and real guiding, the user is responsible for the progress of the task, and particularly benefits from the accurate positioning of the tool (cobot end-effector / circular saw) offered by the passive guiding system (computer-generated guide / straight edge system), in the form of haptic feedback. This concept has been used in industrial and surgical comanipulation applications [Becker et al., 2013], [Dumora, 2014] and teleoperation [Abbott, 2005], [Bowyer and y Baena, 2013]. In our work we are interested in generating paths for the robot that can be used as *virtual guiding fixtures* in a comanipulation context, see Fig.1.

Fig. 1: Experimental setup for user study with a 3-DOF ISYBOT collaborative robot.

One issue in using virtual guides is that their definition often requires robot programming skills and task's modeling. So the usefulness of the method is limited in presence of changing constraints or where multiple tasks need to be solved sequentially but in an undefined order. In the previous work [Raiola et al., 2015], we proposed a probabilistic framework to allow non-expert users to demonstrate new guides by kinesthetic teaching even if already known guides are active. Users may also select the appropriate guide from a set of guides through physical interaction with the robot. This probabilistic framework involves modeling a demonstrated set of guides with Gaussian Mixture Models (GMM) and retrieving a generalized representation of the data set using Gaussian Mixture Regression (GMR). Unfortunately, when working with a comanipulation robot, it could be exhausting to repeat several times the desired movement in order to generate a data set to encode the virtual guides. Indeed, the inertia of the cobot itself (the weight of a cobot is typically

[1]Interactive Robotics Laboratory (LRI), CEA-List, Gif-sur-Yvette, France
[2]LAAS-CNRS, University of Toulouse, CNRS, UPS, Toulouse, France
[3]ENSTA ParisTech, University Paris-Saclay, Palaiseau, France
[4]FLOWERS Team, at INRIA Bordeaux Sud-Ouest

around 15 kg but could go up to 40 kg) and remaining articular friction, can typically disrupt the fluent execution of movements.

For this reason, we propose an iterative approach that allows the worker to locally modify the guides while being assisted by the robot. Only one demonstration without assistance is needed. One contribution of this paper is the use of virtual guiding assistance to program virtual guides iteratively. We demonstrate a concrete implementation of our method using kinesthetic teaching and Akima splines. Our approach enables users to define virtual guides by demonstration. They may also iteratively modify the guides by changing cartesian points or a portion of the guide through physical interaction with the robot.

This paper is organized as follows. After describing the state of the art in Section II, we explain in Section III our virtual guiding fixtures implementation using virtual mechanisms [Joly and Andriot, 1995] and Akima splines [Akima, 1970]. In Section IV we present our iterative virtual guides programming method. We evaluate our approach with a comanipulation robot in Section V and conclude with Section VI.

II. RELATED WORK

A. Virtual guides definition

Virtual guides are used to passively enforce virtual constraints on the movements of cobots, in order to assist the user during a collaborative task. Virtual guides have been first introduced by Rosenberg [Rosenberg, 1993] as *Virtual Fixtures*. The fundamental concept is that virtual fixtures can reduce mental workload, task time and errors during teleoperated manipulation tasks. After Rosenberg's initial work, the use of virtual fixtures has been extended to robotic surgery under the name of *active constraints* [Davies et al., 2006] and to industrial applications in the context of *Intelligent Assist Devices* [Colgate et al., 2003]. Nowadays, virtual fixtures have been featured in several different works, but unfortunately "there is currently no definitive concept which unifies the field" [Bowyer et al., 2014] because of the different definitions, applications and implementation methods. Generally, virtual fixtures have been used in teleoperation [David et al., 2014], [Xia et al., 2013] or comanipulation contexts [Lin et al., 2006], [Dumora, 2014]. The type of assistance offered by the virtual fixtures can vary among different definitions [Abbott et al., 2007], but in general, they are either used to guide the user along a task-specific pathway [Marayong et al., 2003], [Burghart et al., 1999], [Bettini et al., 2004] or to limit the user to move the robot within a safe region [Abbott and Okamura, 2003]. In the first case, we refer to the virtual fixtures as *virtual guiding fixtures* or simply as *virtual guides*.

The particular implementation of virtual guides we use is based on the work presented by Joly [Joly and Andriot, 1995], where a passive virtual mechanism is connected to the robot end-effector by a spring-damper system in a teleoperation context. Virtual mechanisms have also been used in [Pezzementi et al., 2007], where they are called *proxies*. Virtual guides may also be implemented by using anisotropic admittances to attenuate the non-preferred user force components [Marayong et al., 2003], [Bettini et al., 2004]. These methods require sensing external inputs, such as the force or the velocity applied by the user on the robot end-effector. This is not required in our control scheme.

B. Virtual guides construction

There are many possible solutions to construct virtual guides. Usually, the way to create them is strictly related to the goals of the final application. In general, virtual guides have often been limited to pre-defined geometric shapes [Marayong et al., 2003] or combinations of shapes [Aarno et al., 2005], [Kuang et al., 2004] or well-defined geometric models [Joly and Andriot, 1995], [Dumora, 2014] or defined through high level tasks models [Xia et al., 2013]. In [David et al., 2014], David &. al. proposed a supervisory control system using virtual guides to speed up a disk-cutter insertion process. Virtual guides are created on the fly into a physical engine using linear interpolations. In this teleoperation context, environment modeling is also required. One drawback of this method is the possible position errors induced by the fact that the model may be mis-referenced with the reality. In a comanipulation context, it would be more natural to program virtual guides in the real workspace rather than in a simulated one. Therefore, Programming by Demonstration (PbD) [Calinon et al., 2010] appears as a promising strategy to program robots in a fast and simple way when the task is known by the user. During PbD, the operator can directly manipulate the robot end-effector to teach a desired movement.

Generating guides from demonstrations has been explored by Aarno & al. [Aarno et al., 2005]. Their adaptive approach used Hidden Markov Models to model and detect optimum guides obtained by demonstration and represented as a sequence of linear guides. In previous work [Raiola et al., 2015], we proposed a framework for multiple probabilistic virtual guides where kinesthetic teaching and GMM were used to implement virtual guiding fixtures. This probabilistic framework involves modeling a demonstrated set of guides with GMM and retrieving a generalized representation of the data set using GMR. Unfortunately, a compromise must be made between the number of demonstrations (time and effort demanding) and the level of information in the training data.

C. Virtual guides modification

The virtual guides obtained with the mentioned PbD approaches cannot be modified online. If the task changes, the operator has to do a new set of demonstrations with the robot to obtain a new task representation. To overcome this flexibility drawback, in the approaches presented by Rozo et al. [Rozo et al., 2014] and Aarno et al. [Aarno et al., 2005], the robot is able to automatically adapt. However, in the first approach, this is done for tasks where initial and end points are more relevant than the trajectory itself. In the second, the fixtures are made flexible and adaptive by decomposing the trajectory into straight lines. The probability that

the user is following a certain trajectory is estimated and used to automatically adjust the compliance of the virtual guide. Similarly, we previously explored in [Raiola et al., 2017] an iterative method combining an incremental GMM training and clustering, but, given the probabilistic nature of GMM, multiple complete demonstrations are still needed to correctly modify the guides.

From another point of view, the authors of [Boy et al., 2007] introduced the concept of *collaborative learning* to design ergonomic virtual guides to a tricycle cobot and adapt motion to changes in the environment. PbD is used to teach the cobot a path to follow. A dedicated GUI path editor is provided for offline definition and modification of guide paths. In the same manner, it was suggested in [Mollard et al., 2015] to improve interaction in a PbD context by including a GUI in the programming loop to show the learned information. A relevant difference between the approaches of Boy et al. [Boy et al., 2007] and Mollard et al. [Mollard et al., 2015] is that the second approach intends to optimize the learning of a task aimed to be automatically reproduced by the robot, while the first approach uses the operator not only as a part of the teaching phase but as a part of the task execution. Thus, motion guidance is not implemented in [Mollard et al., 2015].

In our approach, we suggest to assist the user throughout the teaching process. A first virtual guide assistance is created using PbD with only one demonstration or a preprogrammed trajectory. The virtual guide is defined by an Akima spline [Akima, 1970], created using the 3D points of the demonstrated or preprogrammed trajectory. At this stage, the user is able to test the guide assistance by haptic feedback and modify it online while the assistance is active, by changing a single point or a portion of the virtual guide. This is possible because Akima splines allows local deformation of the control points.

Under this perspective, the work in [Martin Tykal and Kyrki, 2016] appears to be much closer to our motivations. This method was proposed to ease kinesthetic teaching by assisting the user during teaching using virtual tool dynamics [Kosuge et al., 1995]. However, the assistance is gradually increased based on the accumulated demonstrations. Therefore, several demonstrations are still needed to refine the task before getting the correct assistance. Moreover, after each iteration, it is the robot who chooses an assistance and not the operator who decides where to refine the trajectory, which might be counterintuitive to the user. One of the advantages of our method is that the operator is the master of the teaching and decides when and where a trajectory modification has to be done.

III. VIRTUAL GUIDES IMPLEMENTATION

We propose a framework for virtual guides programming and demonstrate a concrete implementation of virtual guides using kinesthetic teaching. Our approach enables non-expert user to design virtual guides by demonstration while being assisted by them. Next we describe our particular implementation of virtual guides using the virtual mechanisms proposed by Joly [Joly and Andriot, 1995] and Akima splines [Akima, 1970].

A. Control law

The idea of virtual guiding fixtures is to provide a clear and simple perception of the desired behavior of the cobot. To do so, we use the concept of virtual mechanisms introduced in [Joly and Andriot, 1995] where the cobot end-effector is virtually connected to a virtual mechanism through a spring-damper system. The result is a confined motion of the cobot end-effector if the virtual mechanism possesses less number of degrees of freedom (DOFs) than the cobot.

We use a 1 DOF virtual mechanism, implemented as a desired 3D path (see Fig.2). The cartesian position and velocity of the virtual mechanism and the cobot end-effector are described by $\{x_{vm}, \dot{x}_{vm}\}$ and $\{x, \dot{x}\}$, respectively. The virtual mechanism is connected to the cobot end-effector with a *spring-damper* system, which corresponds to a *proportional-derivative* controller whose coupling gains are the stiffness K and the damping B. The current position of the virtual mechanism is described in its parameterized space by the parameter s_{vm}. The evolution of the virtual mechanism is described by $\dot{s}_{vm}$. The direct geometric and kinematic models of the virtual mechanism are defined by L_s and J_s, respectively.

$$x_{vm} = L_s(s_{vm}) \tag{1}$$

$$\dot{x}_{vm} = J_s \dot{s}_{vm} \tag{2}$$

where J_s is the virtual mechanism's Jacobian.

$$J_s = \frac{\partial x_{vm}}{\partial s_{vm}}$$

The force F_c applied by the spring-damper system to the cobot is given by:

$$F_c = K(x_{vm} - x) + B(\dot{x}_{vm} - \dot{x}) \tag{3}$$

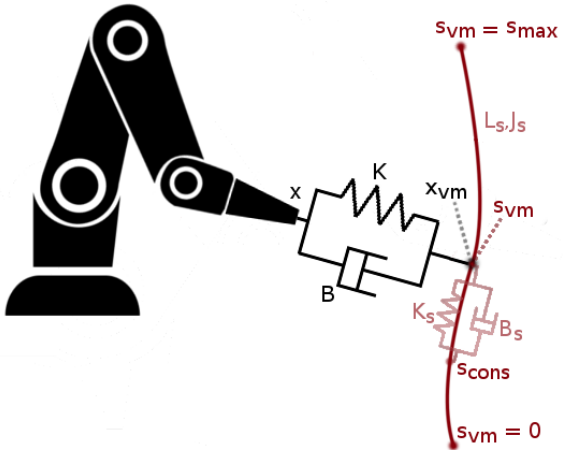

Fig. 2: Lateral view of the physical analogy of a virtual mechanism. The desired path is illustrated in red. The point x of the cobotic system is linked to the point x_{vm} of the virtual mechanism by a spring-damper system. s_{vm} is the current position of the virtual mechanism.

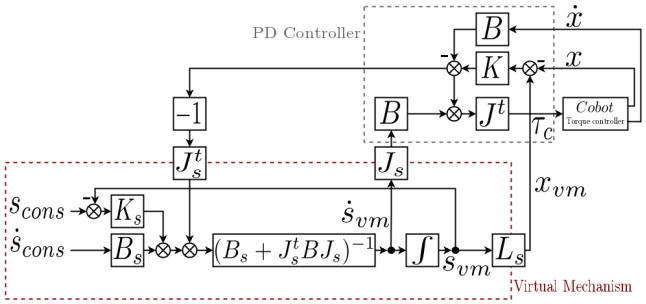

Fig. 3: Control law scheme of a 1-DOF virtual mechanism.

The cobot's Jacobian is given by J:

$$J = \frac{\partial x}{\partial q}$$

where q represents the cobot's articular position.

The torque applied to the cobot's joints is described by τ_c.

$$\tau_c = J^t F_c \tag{4}$$

It is possible to specify a stiffness-damping coupling (K_s and B_s) of the virtual mechanism to a reference position s_{cons} along the desired path. It is then possible to create virtual boundaries at the extremities of the path by applying to s_{cons} the following law:

- If $s_{vm} \in (0, s_{max})$ then $s_{cons} \leftarrow s_{vm}$ and $\dot{s}_{cons} \leftarrow \dot{s}_{vm}$.
- If $s_{vm} > s_{max}$ then $s_{cons} \leftarrow s_{max}$ and $\dot{s}_{cons} \leftarrow 0$.
- If $s_{vm} < 0$ then $s_{cons} \leftarrow 0$ and $\dot{s}_{cons} \leftarrow 0$.

The behavior of the virtual mechanism's impedance is given by:

$$T_{vm} = K_s(s_{cons} - s_{vm}) + B_s(\dot{s}_{cons} - \dot{s}_{vm}) \tag{5}$$

The equilibrium of the forces applied to the virtual mechanism is given by:

$$J_s^t F_c = T_{vm} \tag{6}$$

Using equations (2), (3), (5) and (6), we obtain:

$$J_s^t(K(x_{vm} - x) + B(J_s\dot{s}_{vm} - \dot{x})) = -B_s\dot{s}_{vm} + K_s(s_{cons} - s_{vm}) + B_s\dot{s}_{cons} \tag{7}$$

By solving equation (7) with respect to $\dot{s}_{vm}$, we obtain a first order dynamical system that expresses the evolution of the virtual mechanism s_{vm}:

$$\dot{s}_{vm} = (B_s + J_s^t B J_s)^{-1}(-J_s^t(K(x_{vm} - x) - B\dot{x})) + K_s(s_{cons} - s_{vm}) + B_s\dot{s}_{cons} \tag{8}$$

s_{vm} can be determined at every instant by integrating the controller state equation (8) in real time. From equations (1), (2), (3), (4), (5) and (8) we obtain the control law scheme presented in (Fig. 3).

The gains specifications (K and B) of the coupling are independent from the guide specification (L_s, K_s and B_s). Gains tunning for (K and B) is similar to the tunning of a cartesian PD position loop.

The passivity[1] of the virtual mechanism controller is proven by Joly, in [Joly and Andriot, 1995], by using the mechanical analogy of the system and studying the energy dissipation. Moreover, it was proven by Hogan [Hogan, 1988] that the passivity of the system guarantees the stability of the controlled system when it interacts with any passive environment, including a human operator.

B. Virtual guides construction

In the first teaching phase, the operator can use a pre-programmed path or show to the cobot the desired trajectory. The user manually moves the cobot end-effector and records points by demand or continuously with a determined sampling time[2]. In both cases, the cartesian position x_{vm} of the cobot end-effector is stored as a list of N points: $\{x_i, y_i, z_i\}_{i=0:N-1}$.

Tool orientation programming is not addressed here, but will be published in a future article as an extension of this work.

We reconstruct the virtual guiding fixture from the stored points by using a local cubic polynomial Akima interpolation [Akima, 1970]. This method is a continuously differentiable sub-spline interpolation. It is built from piecewise third order polynomials, where only data from the next and previous two neighbor points are used to determine the coefficients of the interpolation polynomial. The slope of the curve is locally determined at each given point by the coordinates of five points centered on the studied point. This spline type creates a smooth curve between the recorded points and always passes directly through them.

Some of the advantages of this interpolation method are:

- There is no need to solve large equation systems. It is therefore computationally very efficient.
- Akima spline interpolation reduces oscillatory effects.
- Local changes do not affect the interpolation beyond neighbor points.
- Akima spline points are intuitive to use in the trajectory modification.

In our implementation, the direct geometric model L_s of the virtual mechanism (see Eq.1) is defined by the Akima spline interpolation. Thus, the direct kinematic model J_s (see Eq.2) is defined by the spline's derivate function.

Virtual guide parameterization:

If we use time as the parameter for the spline, the Jacobian J_s will correspond to the virtual mechanism's velocity. Since the trajectory is shown by demonstration, this velocity is variable and could be null. Jacobian variations could affect the user's interaction with the virtual mechanism (see Eq.8). In order to guarantee a normalized virtual mechanism's Jacobian, it is desirable to evaluate the Akima spline at points based on its arc-length instead of its recording sampling time. For that matter, we propose to separate spacial and temporal aspects of the trajectory. Let t be the time, s be the path-length curvilinear parameter (which corresponds to s_{vm} in Fig.2); M the 3D vector containing the Akima spline points; $f : \mathbb{R} \to \mathbb{R}^3$, $f : t \to M$, the Akima spline parametrized

[1] A system is considered passive when it does not provide more energy than it has received.

[2] The interested reader on key points definition for PbD trajectory learning can refer to [Vakanski et al., 2012].

with time; $g : \mathbb{R} \to \mathbb{R}$, $g : s \to t$ the transformation function between path-length curvilinear parameterization and time. We approximate the computation of s by:

$$s_i = \sum_{j=1}^{i-1} \|M_j - M_{j-1}\|$$

where $M_j = \{x_j, y_j, z_j\}$.

The Akima spline is now defined as a composition of the curve f parameterized with time and the space transformation g as: $f(g(s)) = f \circ g(s)$.

IV. Iterative virtual guides programming

When asking a user to perform several times the ideal path he has in mind, there may be many variations. These variations could exist due to the presence of friction and gravity forces that the user must compensate, the poor repeatability inherent to human gestures, the variability in the task and, often, simply concentration errors on a trajectory portion. For these reasons and the ones enlightened in Section II, we suggest that the user program the virtual guides by iteratively modifying them while being assisted by the cobot. In this section, we present a local guide's refinement method. The refinement is done directly on the workspace by manually manipulating the cobot end-effector to show a new portion of the guide. During local refinement, the operator may be more focused than during the previous demonstration since he is able to demonstrate again only the portion and not the entire trajectory. The main advantage of this approach is that the worker is assisted throughout the iterative teaching phase and only one entire demonstration of the task is needed.

A. Scaled force control

In order to show the new portion of trajectory while the virtual guide is active, the user needs to momentarily escape the guide. To this matter, we use the concept of soft virtual guides and force scaling presented in [Nolin et al., 2003], [Raiola et al., 2015] to allow the user to go off the path and locally modify the virtual guide. When the user tries to go off the path, the virtual guide controller's force fade proportionally with the distance between the guide's position x_{vm} and the current position of the robot end-effector x. The user would feel an attractive force F when escaping or approaching the guide, up to a defined distance d_{max}.

$$F = \beta(X) F_c$$

For computational efficiency purposes, $\beta(X)$ is defined as a 4th degree polynomial with $X = \frac{\| x - x_{vm} \|}{d_{max}}$, and null when $X > 1$:

if $X <= 1$ **then**
  $\beta(X) = x^4 - 2x^2 + 1$
else
  $\beta(X) = 0$
end if

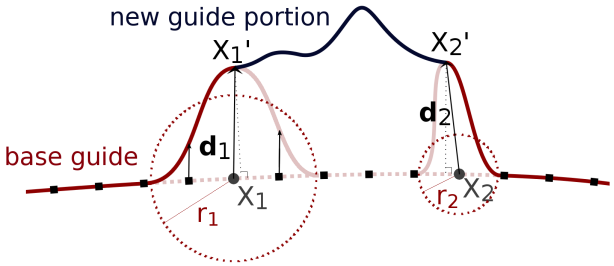

Fig. 4: Local refinement applied to the constraint points X_1 and X_2 lying on the base guide. $X_1^{'}$ and $X_2^{'}$ are the initial and final points of the new guide portion.

The parameter d_{max} can be tuned manually to modify the basin of attraction of the virtual guide.

A more distinctive feature of this function is the possibility to go back to the base guide intuitively without needing to change the control mode.

B. Local guide's refinement

Initial and final positions of the partial demonstration do not always match a control point of the guide. To merge the new portion with the rest of the guide, we propose a method to modify the closest points on the base guide for matching the first and last points on the new guide portion. We got inspiration from the SCODEF (Simple Constrained Object Deformation) concept. This method was introduced by Borrel, [Borrel and Rappoport, 1994] in the field of geometric modeling and interactive shape edition, for producing controlled spatial deformations. Our contribution is to implement a simplified version of the SCODEF method to locally modify the virtual guides, reconstructed as Akima splines.

Let X_1 and X_2 be the constraint points of the current guide. $X_1^{'}$ and $X_2^{'}$ are the initial and final points of the new guide portion, respectively (see Fig.4). The constraint points are defined as the two base guide's points which are closest to $X_1^{'}$ and $X_2^{'}$, respectively. We obtain X_1 and X_2 by calculating the distance from $X_1^{'}$ and $X_2^{'}$ to the base guide's control points. Then we select the two control points of minimum distance. Displacement vectors $\vec{d}_1$ and $\vec{d}_2$ are determined between X_1 and $X_1^{'}$, and between X_2 and $X_2^{'}$. A radius of influence r must be defined for both constraints. We choose the radius proportional to the magnitude of the displacement vector.

$$\vec{d}_1 = \overrightarrow{X_1 X_1^{'}}; \qquad r_1 = \alpha_1 \| \vec{d}_1 \|$$
$$\vec{d}_2 = \overrightarrow{X_2 X_2^{'}}; \qquad r_2 = \alpha_2 \| \vec{d}_2 \|$$

The radius $r1$ and $r2$ allow a more intuitive control of the deformation. The only parameters to tune are α_1 and α_2, where a compromise must be done between the smoothness of the curve and the area of influence of the deformation, i.e, the neighbor points that will be deformed. We define a local deformation function F centered at the constraint points and decreasing to zero for points beyond the radius. In order to obtain a smooth displacement of the spline points, we determine a fourth degree polynomial as

the deformation function $F(x)$, where $x = \frac{(s_{vm,i} - s_{vm,0})}{r}$. $\{s_{vm,i}\}_{i=0:N-1}$ represent the curvilinear abscissa parameter of the Akima spline, and N is equal to the number of interpolation points.

$$f(x) = x^4 - 2x^2 + 1$$

The deformation function $F(x)$ is though defined as:

$$F(x) = f(x)\vec{d}$$

The deformation function $F(x)$ is applied to the X_1 and X_2 neighbor points within the radiuses of influence r_1 and r_2, in order to obtain the new control points that would now include $X_1^{'}$ and $X_2^{'}$. These new guide's control points are stored in a vector defined by the modified base guide's control points before X_1 and after X_2, along with the new guide portion's control points. Finally, we perform a new Akima spline interpolation to define the new virtual guide.

V. EXPERIMENTAL EVALUATION

The following experiment was conducted with a 3-DOF ISYBOT collaborative robot (see Fig.1) and a sander. The general task was to use the cobot to simulate a sanding task. The simulated task consisted in using the tool to clean a metal sheet while following a sweeping trajectory defined by 2 red erasable marks. We used black tape to mark the trajectory in order to always draw the same red marks. It was necessary to choose a simple task to minimize the impact of different skill levels of the user. However, the system could be applied to more complex scenarios.

Two tasks were studied: *Case A* consisted on the cleaning task explained before and *Case B* consisted in performing the same task with an obstacle blocking the trajectory. In both cases, we compared two assistance modes: *Mode A* uses gravity compensation and *Mode B* uses virtual guiding assistance. In *Mode A*, the user was able to move the robot freely. For *Mode B*, we asked an expert user to program the virtual guide.

A. Programming virtual guides by an expert user

To program a virtual guide for *Mode B - Case A*, we used a function that takes two nonadjacent vertices of a rectangle and generates a set of points describing a sweeping trajectory filling the rectangle. These vertices were shown by demonstration to the robot and recorded using a button located on the robot's third axis (cf. video supplement[3]). With this set of points $\{x_i, y_i, z_i\}_{i=0:N-1}$, we generated a virtual guide as explained in Section III (see Fig.5,i). For *Mode B - Case B*, it was not possible to entirely use the previous guide since there was an obstacle blocking the trajectory. We asked the expert user to modify the guide using the approach explained in Section IV (cf. video supplement). While the guide was active, the user was able to move along the trajectory. When the user arrived near the obstacle, he/she

[2]Back-drivable cobot with screw-and-cable transmission, weight = 40 kg and payload = 8 kg. More details at: http://www.sybot-industries.com/

[3]The HD version of the video is available at: `goo.gl/cSnJht`

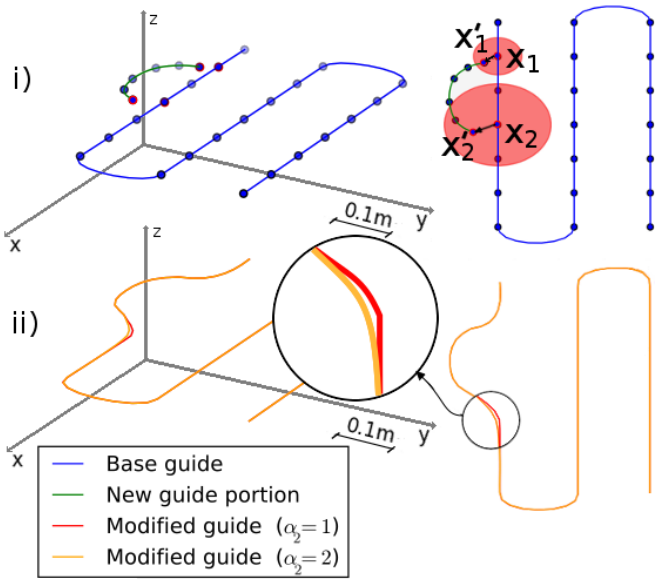

Fig. 5: i) Base guide and new guide portion. The areas of influence are represented by the red circles which are centered at the constraint points X_1 and X_2, lying on the base guide. Each radius of influence is α times proportional to the magnitude of the correspondent displacement vector, as explained in Section IV. Here, $\alpha_1 = 1$ and $\alpha_2 = 2$. $X_1^{'}$ and $X_2^{'}$ are the initial and final points of the new guide portion. ii) Resulting virtual guide after local refinement. The choice of each α has an impact on the final curve. In our case, we compare $\alpha_2 = 1$ and $\alpha_2 = 2$ (α_1 is fixed to 1). $\alpha_2 = 2$ allows to reach another key point of the base guide, thus improving smoothness of the resulting curve.

was able to escape the guide as presented in Section IV.A. Then, the new points of the partial modification were shown by demonstration to the robot and recorded using a button located on robot's third axis (see Fig.5,i). After the last point was recorded, the upper black button launched the refining algorithm and a new guide was created (see Fig.5,ii). For the virtual guide assistance, the stiffness was set as $K = kI$, with $k = 10000\,N/m$ and $I = [1;1;1]$. The damping was set as $B = bI$, with $b = 400\,N/ms^{-1}$ and $I = [1;1;1]$. Sampling time is $1ms$. During the virtual guide's modification, we set: $d_{max} = 0.01m$, $\alpha_1 = 1$ et $\alpha_2 = 2$. These values were chosen through trial-and-error to meet the conditions described in Section IV. An example of the influence of the choice of α_1 et α_2 is shown in Figure 5,ii.

Our approach allowed the expert user to modify the first guide while being assisted by the it, in order to adapt to a change of environment. In this case of application, only the demonstration of two points and of a portion of trajectory were needed to obtain two guides. Without our method, the user would have had to do an entire demonstration with the cobot in order to create a new guide to avoid the obstacle.

B. User study

We designed a pilot study to observe how novice users perceived the virtual guide assistance and performed with

	Mode A		Mode B			
Question	**Mean**	**SD**	**Mean**	**SD**	**F**	**p-value**
Do you think that you performed the task well?	5.21	1.19	5.89	0.91	F(1,12)=9.7028	0.0089
Did you feel you performed the task precisely?	4.82	1.19	5.64	1.06	F(1,12)=11.207	0.0058
Do you think the robot was helpful during the task execution?	3.82	1.58	5.29	1.65	F(1,12)=4,6126	0.0529
Do you think the robot was easy to work with?	5.11	1.26	5.39	1.37	F(1,12)=,50585	ns

Table I
Survey results of the user study in *Mode A* and *Mode B*

the cobot. We recruited 14 participants from our research laboratory (between 22 and 33 years old, 5 females). Nine participants stated they had prior experience with robots, ranging from robotic courses to hands-on experience with industrial robots.

Three hypotheses were tested:

- H1: Virtual guides assistance reduces the task's execution time.
- H2: Virtual guides assistance improves task's performances.
- H3: Virtual guides are easy to use.

All participants were asked to perform the tasks *Case A* and *Case B* (ie. cleaning task; cleaning task with an obstacle), in both modes, *Mode A* and *Mode B* (ie. gravity compensation; virtual guide assistance), resulting to four test conditions. The four test conditions were presented in a randomized order to avoid training effects. For each condition, the participants were asked to perform the task 3 times in a row (Repetitions). At the beginning of each condition, the *Case* and the *Mode* were presented to the participants and they were able to familiarize themselves with the system. When a condition was completed, it was asked to the participants to fill a post-condition survey (Likert-scale survey with a rating from 1 to 7, with 1 as strong disagreement and 7 as strong agreement, the same for each condition). In total, the participants performed $12 = 2\times2\times3$ (Case*Mode*Repetition) cleaning tasks.

To validate our hypotheses, we recorded the participants' times of execution of the 12 tasks and the answers to the four post-condition surveys. The measures we were interested in are the following:

- Execution time, to validate H1 and H3.
- Observed collisions in case B, to validate H2.
- Survey results, to validate H2 and H3.

These results are summarized in Fig. 6, Fig. 7 and Table I. We performed a repeated-measure ANOVA on three factors: (1) Cases, (2) Modes, and (3) Repetitions. Participants were grouped by their habit to use a robot. Post hoc pairwise comparisons were computed using non-pooled error terms (i.e., by computing separate paired-samples t tests; sequentially acceptive step-up Benjamini [Benjamini and Hochberg, 1995] procedure, with an alpha level of .05.

User study results:

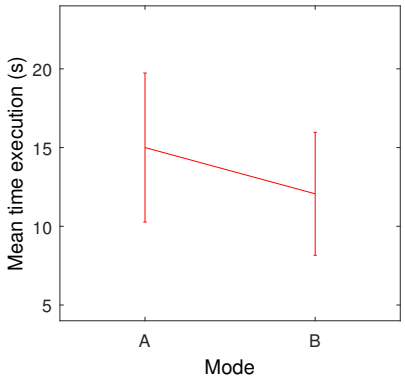

Fig. 6: Task's execution time. Comparison between *Mode A* (gravity compensation) and *Mode B* (virtual guiding).

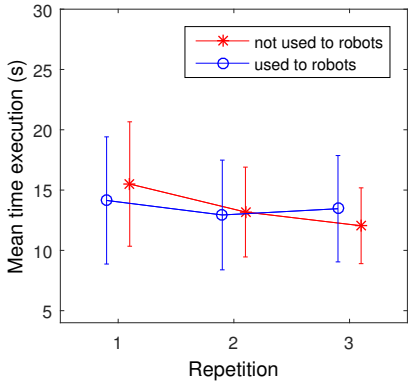

Fig. 7: Task's execution time. Comparison between repetitions for participants used and not used to work with robots.

We found a significant main effect of the Modes on the execution time of the participants ($F(1,12) = 14.78$; $p < .01$). In *Mode A*, participants were slower than with in *Mode B* (*Mode A*: $M = 15, SD = 4.73$; *Mode B*: $M = 12.06, SD = 3.90$) (see Fig.6). This indicates that the virtual guides reduced the execution time. This validates H1. We also found a significant main effect of the Repetitions on the execution time of the participants ($F(2,24) = 14.79; p < .001$). Post hoc analyses showed that the first repetition was longer than the second ($p < .025$) and the third repetitions ($p < .001$). This confirms there is a training effect. Moreover, there is a significant effect of the interaction between repetitions and one's experience with a robot ($F(2,24) = 5.76, p < .01$) (see Fig.7). Post hoc analyses showed that the first repetition was longer than the second ($p < .025$) and the third repetitions ($p < .025$) only in the group of participants not used to work

with robots. This indicates us that the participants who had never worked with a robot before had a greater improvement in their execution time than those who had. These results support the hypothesis H3.

For clarity reasons, the answers to four questions of our user study for each *Mode* are summarized in Table I. With this survey, we observed a significant effect on the *Mode*. In *Mode B*, ie. with the virtual guide, participants found that: (1) they performed better the task; (2) they felt more precise in the execution in the task; (3) the robot was more helpful than in *Mode A*. Moreover, for *Case B*, collisions only occurred when the users were not assisted. These results validate H2.

The results of this study are similar/comparable to previous work [Lin et al., 2006], [Vozar et al., 2015] and confirm the interest of virtual guiding.

VI. Conclusion

We presented a novel approach for using virtual guiding assistance to program virtual guides. We proposed a novel implementation of virtual guides using virtual mechanisms and Akima splines. Our approach enables users to create virtual guides by demonstration. Users may also iteratively modify the guides by modifying a portion of the guide through physical interaction with the cobot. We suggested to use a scaled force control to escape the active guide in order to modify it. The experimental evaluation of the system with an expert user showed an application of our approach to a sanding task with a cobot where the task changes and the virtual guides must be reprogrammed by the operator. Furthermore, the user study demonstrates that these guides improve the performance and the execution time of comanipulation tasks. Also, the virtual guide assistance is perceived as helpful by novice users.

In our future work, we will perform an user study on the proposed approach for virtual guides iterative modification, in order to evaluate the intuitiveness of our method. An important enhancement would extend the programming approach to a 6D assistance, where orientation would be taught by demonstration. Moreover, it would be interesting to study if splitting the orientation from the position trajectory programming would lead to an easier and qualitatively better experience for the users.

References

[Aarno et al., 2005] Aarno, D., Ekvall, S., and Kragic, D. (2005). Adaptive Virtual Fixtures for Machine-Assisted Teleoperation Tasks. In *ICRA*.

[Abbott et al., 2007] Abbott, J., Marayong, P., and Okamura, A. (2007). Haptic virtual fixtures for robot-assisted manipulation. *Springer Tracts in Advanced Robotics*.

[Abbott, 2005] Abbott, J. J. (2005). *Virtual Fixtures for Bilateral Telemanipulation*. PhD thesis.

[Abbott and Okamura, 2003] Abbott, J. J. and Okamura, A. M. (2003). Virtual fixture architectures for telemanipulation.

[Akima, 1970] Akima, H. (1970). A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures. *J. ACM*.

[Becker et al., 2013] Becker, B. C., MacLachlan, R. A., Lobes, L. A., Hager, G. D., and Riviere, C. N. (2013). Vision-Based Control of a Handheld Surgical Micromanipulator With Virtual Fixtures. *IEEE Transactions on Robotics*.

[Benjamini and Hochberg, 1995] Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*.

[Bettini et al., 2004] Bettini, A., Marayong, P., Lang, S., Okamura, A. M., and Hager, G. D. (2004). Vision assisted control for manipulation using virtual fixtures. In *IROS*.

[Borrel and Rappoport, 1994] Borrel, P. and Rappoport, A. (1994). Simple Constrained Deformations for Geometric Modeling and Interactive Design. *ACM*.

[Bowyer et al., 2014] Bowyer, S. A., Davies, B. L., and y Baena, F. R. (2014). Active constraints/virtual fixtures: A survey. *IEEE Transactions on Robotics*.

[Bowyer and y Baena, 2013] Bowyer, S. A. and y Baena, F. R. (2013). Dynamic frictional constraints for robot assisted surgery. In *World Haptics Conference (WHC)*.

[Boy et al., 2007] Boy, E. S., Burdet, E., Teo, C. L., and Colgate, J. (2007). Investigation of Motion Guidance With Scooter Cobot and Collaborative Learning. *IEEE Transactions on Robotics*.

[Burghart et al., 1999] Burghart, C., Keitel, J., Hassfeld, S., Rembold, U., and Woern, H. (1999). Robot controlled osteotomy in craniofacial surgery. In *Proceedings of the 1st International Workshop on Haptic Devices in Medical Applications*.

[Calinon et al., 2010] Calinon, S., D'halluin, F., Sauser, E., Caldwell, D., and Billard, A. (2010). Learning and Reproduction of Gestures by Imitation. *IEEE Robotics Automation Magazine*.

[Colgate et al., 2003] Colgate, J., Peshkin, M., and Klostermeyer, S. (2003). Intelligent assist devices in industrial applications: a review. In *IROS*.

[Colgate et al., 1996] Colgate, J. E., Edward, J., Peshkin, M. A., and Wannasuphoprasit, W. (1996). *Cobots: Robots For Collaboration With Human Operators*.

[David et al., 2014] David, O., Russotto, F.-X., Da Silva Simoes, M., and Measson, Y. (2014). Collision avoidance, virtual guides and advanced supervisory control teleoperation techniques for high-tech construction: framework design. *Automation in Construction*.

[Davies et al., 2006] Davies, B., Jakopec, M., Harris, S. J., Baena, F. R. Y., Barrett, A., Evangelidis, A., Gomes, P., Henckel, J., and Cobb, J. (2006). Active-constraint robotics for surgery. *Proceedings of the IEEE*.

[Dumora, 2014] Dumora, J. (2014). *Contribution à l'interaction physique homme-robot : Application à la comanipulation d'objets de grandes dimensions*. PhD thesis.

[Hogan, 1988] Hogan, N. (1988). On the stability of manipulators performing contact tasks. *IEEE Journal on Robotics and Automation*.

[Joly and Andriot, 1995] Joly, L. and Andriot, C. (1995). Imposing motion constraints to a force reflecting telerobot through real-time simulation of a virtual mechanism. In *IROS*.

[Kosuge et al., 1995] Kosuge, K., Itoh, T., Fukuda, T., and Otsuka, M. (1995). Tele-manipulation system based on task-oriented virtual tool. In *ICRA*.

[Kuang et al., 2004] Kuang, A., Payandeh, S., Zheng, B., Henigman, F., and MacKenzie, C. (2004). Assembling virtual fixtures for guidance in training environments. In *HAPTICS*.

[Lamy, 2011] Lamy, X. (2011). *Conception d'une interface de pilotage d'un Cobot*. PhD thesis.

[Lin et al., 2006] Lin, H. C., Mills, K., Kazanzides, P., Hager, G. D., Marayong, P., Okamura, A. M., and Karam, R. (2006). Portability and applicability of virtual fixtures across medical and manufacturing tasks. In *ICRA*.

[Marayong et al., 2003] Marayong, P., Li, M., Okamura, A. M., and Hager, G. D. (2003). Spatial motion constraints: theory and demonstrations for robot guidance using virtual fixtures. In *ICRA*.

[Martin Tykal and Kyrki, 2016] Martin Tykal, A. M. and Kyrki, V. (2016). Incrementally assisted kinesthetic teaching for programming by demonstration.

[Mollard et al., 2015] Mollard, Y., Munzer, T., Baisero, A., Toussaint, M., and Lopes, M. (2015). Robot programming from demonstration, feedback and transfer. In *IROS*.

[Nolin et al., 2003] Nolin, J. T., Stemniski, P. M., and Okamura, A. M. (2003). Activation cues and force scaling methods for virtual fixtures. In *HAPTICS*.

[Pezzementi et al., 2007] Pezzementi, Z., Hager, G. D., and Okamura, A. M. (2007). Dynamic guidance with pseudoadmittance virtual fixtures. In *ICRA*.

[Raiola et al., 2015] Raiola, G., Lamy, X., and Stulp, F. (2015). Co-manipulation with multiple probabilistic virtual guides. In *IROS*.

[Raiola et al., 2017] Raiola, G., Sánchez Restrepo, S., Chevalier, P., Rodriguez-Ayerbe, P., Lamy, X., Tliba, S., and Stulp, F. (2017). Co-manipulation with a library of virtual guiding fixtures. *Autonomous Robots, Special issue on Learning for Human-Robot Collaboration*.

[Rosenberg, 1993] Rosenberg, L. (1993). Virtual fixtures: Perceptual tools for telerobotic manipulation. In *IEEE Virtual Reality Annual International Symposium*.

[Rozo et al., 2014] Rozo, L., Calinon, S., and Caldwell, D. (2014). Learning force and position constraints in human-robot cooperative transportation. In *ROMAN*.

[Vakanski et al., 2012] Vakanski, A., Mantegh, I., Irish, A., and Janabi-Sharifi, F. (2012). Trajectory Learning for Robot Programming by Demonstration Using Hidden Markov Model and Dynamic Time Warping. *SMC*.

[Vozar et al., 2015] Vozar, S., Léonard, S., Kazanzides, P., and Whitcomb, L. L. (2015). Experimental evaluation of force control for virtual-fixture-assisted teleoperation for on-orbit manipulation of satellite thermal blanket insulation. In *ICRA*.

[Xia et al., 2013] Xia, T., Léonard, S., Kandaswamy, I., Blank, A., Whitcomb, L. L., and Kazanzides, P. (2013). Model-based telerobotic control with virtual fixtures for satellite servicing tasks. In *ICRA*.