Semi-Autonomous Cooperative Tasks in a Multi-Arm Robotic Surgical Domain

Miha Deniša[1,2], Kim Lindberg Schwaner[2], Iñigo Iturrate[2] and Thiusius Rajeeth Savarimuthu[2]

***Abstract*— This paper addresses collaborative multi-arm task execution in a robot-aided surgical domain. The proposed approach aims to assist the surgeon by recognizing their intent and autonomously controlling some of the robotic arms. We lean on Learning from Demonstration paradigm, where knowledge is gained by observing humans performing a task. Example cooperative movements are gained in a leader/follower scenario with an operator controlling both arms. Example leader arm movements are encoded in a hierarchical database, which consists of a binary tree enhanced with weighted directed graphs. It is used for online leader movement recognition and prediction. Example follower arm movements are encoded as a set of Dynamic Movement Primitives, which are used for online synthesis of appropriate follower movements, based on recognition and prediction of leader movements. The proposed approach is evaluated on a dual arm robotic surgical system, showing that the operator can perform the cooperative task by only controlling the leader arm. The proposed database structure enables recognition and prediction fast enough for real-time task execution, and the novel approach of speed adaptation ensures both arms are in sync regardless of the operators speed of execution.**

I. Introduction

In the past two decades, Robot-Assisted Minimally Invasive Surgery (RMIS) has been on the rise [1]. Systems for RMIS, such as the da Vinci (Intuitive, Inc.), are equipped with three arms for controlling surgical tools and an arm for controlling endoscope movement. These arms and tools are teleoperated by a surgeon using remote controllers. In case the surgeon needs to move the endoscope, or control the third arm, it is necessary to use a foot pedal to make the switch.

In this paper, we expanded on previous work [2], [3] to automate the movement of one surgical tool (the *follower*), based on the movement of another (the *leader*) in a bimanual task. Although we focus on a bimanual task here, the approach could easily scale to a system with more arms. In this way, multiple arms could be put into motion at once, and not just the two being teleoperated. For example, endoscope movement could be automatically controlled based on the movement of two surgical tools, controlled by the surgeon. Our approach involves two elements: recognition of the correct correspondence between leader and follower and generating follower motion.

To generate follower motion, we propose to apply Learning from Demonstration (LfD). LfD has been explored in industrial, service and humanoid robotics as a way to adapt expert knowledge directly from a human teacher [4]–[6]. As teleoperation is the default method of control for surgical robots, gathering demonstrations is straightforward. Once demonstrations data is available, approaches such as Dynamic Movement Primitives (DMPs) [7], [8] can encode robot control policies from as single demonstration, or derive them from a database of multiple demonstrations [9]–[11]. Gaussian mixture models [12], [13], Probabilistic Movement Primitives [14] or Hidden Markov Models (HMMs) [15] can also be used to encode multiple demonstrations.

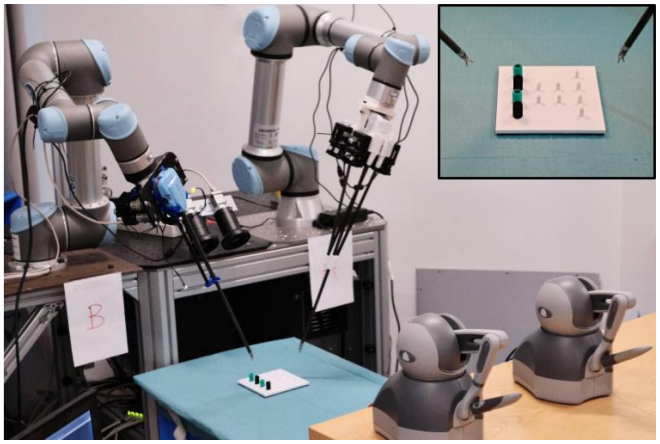

Fig. 1. Overview of the dual arm robotic surgical system used for evaluation. The system consists of two EndoWrist surgical tools (Intuitive, Inc.) mounted on UR5 arms. Two Touch (3D Systems) haptic devices are used to control the robot arms and tools. The peg transfer testbed used for evaluation can be seen in the upper right corner.

Choosing the appropriate follower movement policy to execute based on the leader's state requires two steps: recognition of leader movements and intent, and appropriate collaborative follower movement generation and execution. Such a multi-agent interactive collaborative scenario has been represented using path-map HMMs [16], while hierarchical HMMs have been used to model responsive behaviors [17]. Extensions to movement primitives for collaborative tasks have been proposed in the form of Probabilistic Movement Primitives [14] and Interaction Primitives [18]. One of the drawbacks of the later is unresponsiveness to task changes during recognition [19].

The work most closely related to our proposed approach is that of Yamane et al. [20], [21], where a binary tree database is used to recognize human movement and then synthesize an appropriate robot reaching trajectory.

In the surgical robotics domain, a number of studies have

[1]Humanoid and Cognitive Robotics Lab, Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia; miha.denisa@ijs.si.

[2]SDU Robotics, The Mærsk McKinney Møller Institute, Faculty of Engineering, University of Southern Denmark, Odense, Denmark; {kils,inju,trs}@mmmi.sdu.dk.

investigated LfD and skill transfer [22]–[32] as a means to automate tasks. The peg transfer task is a commonly used training task in surgical robotics because it poses mechanical challenges that can also be observed in real surgical procedures. A number of approaches have previously been suggested for automating variants of said task, either fully or partially, using surgical robot systems [23], [27], [30], [32], [33].

The work in [23], [27], [28] studied human-robot collaborative control of a surgical robot. Transportation motions were automated in [23], [28], while more demanding actions, like grasping and needle insertion, required human input. In [27], [28], the surgical robot assisted the user by providing haptic guidance toward the desired task trajectory.

In previous work, a single hierarchical database was used to synthesize new movements and new compliant movements from a set of example trajectories in a service robot domain [2], [3]. Preliminary evaluation of movement recognition while using hierarchical database was done through comparison to other approaches [19], where follower trajectories were synthesized in a rudimentary way that did not take into account the speed of execution.

To enable execution of cooperative task in the surgical robotics domain, this paper extends previous work to also take into account orientation during recognition and synthesize movements in task space instead of joint space. To ensure synchronisation of leader and follower movements this paper also includes a novel approach for speed adaptation.

While this work is hard to position within the levels of autonomy for robotic surgical systems proposed by [34], the leader arm would fall within level 0 (no autonomy) and the follower arm within level 2 (task autonomy).

The rest of the paper is structured as follows. Section II presents leader movement recognition and appropriate follower movement synthesis. In Section III the implementation and evaluation of the approach on a dual arm robotic surgical system is presented. Concluding remarks are given in the end of the paper.

II. Cooperative Movements

We propose to encode a set of demonstrations of a multimanual task into a database such that, during execution, appropriate follower movements can be generated based on the current movement of the leader. There may be multiple leaders and/or followers. The workflow can be roughly divided into two steps: 1) demonstration and database construction, and 2) execution of the task (see Fig. 2).

The proposed database has two parts: a Hierarchical Database (HDb), which encodes the demonstrated movements of the leader, and a set of corresponding demonstrated movements of the follower, encoded as Dynamic Movements Primitives (DMPs). During the task execution step, the leader is controlled manually and the HDb is searched continuously to recognize and predict the leader movement and automatically generate appropriate follower movement from the corresponding set of DMPs. Speed of the leader is also taken into account.

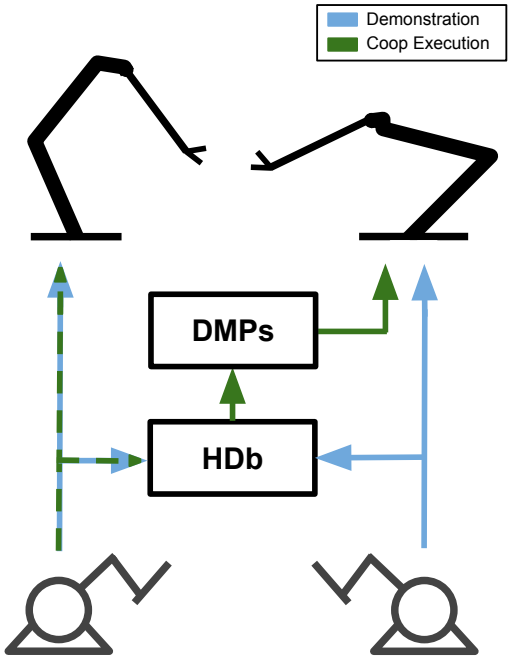

Fig. 2. An outline of the proposed approach. During the demonstration step (blue) each of the two haptic devices are used to move each of the surgical tools. The demonstrated movements of the leader are used to build the Hierarchical Database (HDb). The movements of the follower are encoded as a set of Dynamical Movement Primitives (DMPs). In the execution step (green) only one haptic device is used. It controls the leader, whose movements are also used by the HDb to recognize the leaders intent. This information is then used to synthesize appropriate follower movements based on the corresponding set of DMPs.

The demonstration and database construction step is presented in Section II-A and the execution step is detailed in Sections II-B and II-C.

A. Building the Database

The process of building the database starts with a human operator demonstrating a set of cooperative leader and follower movements. In the case presented in this paper, a dual arm surgical robot is used (see Fig. 1). During demonstration, both arms are controlled by the operator. One arm is denoted the leader, while the second is denoted the follower. This choice can be based on various preferences, e.g. the dominant hand of the operator or the role of the tool in the task.

A set of s cooperative movements $\boldsymbol{D} = \{\boldsymbol{D}_j \mid j = 1, 2, ..., s\}$ is captured from demonstrations. Each demonstration $\boldsymbol{D}_j$ contains n state vectors sampled at discrete times,

$$\boldsymbol{D}_j = \big[\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n\big], \tag{1}$$

where each state vector,

$$\boldsymbol{y}_i = \big[\boldsymbol{y}_i^l, \boldsymbol{y}_i^f\big]^T, \tag{2}$$

includes state vectors $\boldsymbol{y}_i^l$ corresponding to leader arm trajectories, and state vectors $\boldsymbol{y}_i^f$ from the follower arm trajectories. If the captured demonstrations are of different duration, then the number of state vectors n will vary between them. Follower and leader state vectors can be defined differently.

State vectors for end-effector trajectories can, for example, be specified in Cartesian space or in joint space. In the case presented in this paper, leader state vectors $\boldsymbol{y}_i^l$ represent the end-effector positions and orientation of the surgical tools

$$\boldsymbol{y}_i^l = \left[\boldsymbol{p}_i^l, \boldsymbol{q}_i^l\right], \tag{3}$$

where

$$\boldsymbol{p}_i = \left[p_{ix}, p_{iy}, p_{iz}\right] \tag{4}$$

denotes the position in Cartesian space, and

$$\boldsymbol{q}_i = \left[q_{iw}, q_{ix}, q_{iy}, q_{iz}\right] \tag{5}$$

denotes the orientation, represented by a unit quaternion, at sample time t_i. In the case where multiple arms have the role of the leader, state vectors would include positions and orientations of all arms. Leader state vectors also have additional information associated with each of them. This includes the time stamp of the state vector t_i and the index j of the demonstrated set $\boldsymbol{D}_j$ from which the state vector came.

For the presented case, state vectors for follower trajectories,

$$\boldsymbol{y}_i^f = \left[\boldsymbol{p}_i^f, \boldsymbol{q}_i^f, c_i\right], \tag{6}$$

have an additional element c_i, which denotes the grasper opening angle of the follower surgical tool at time t_i.

The demonstrated set of leader movements is then used to build the HDb, while the corresponding follower movements are encoded as DMPs. As mentioned, the correspondence between the leader and follower demonstrations is stored via additional information.

A simplified representation of an HDb can be seen in Fig. 3. In the first step of building the HDb, all leader state vectors $\boldsymbol{y}^l$ belonging to all of the demonstrated movements $\boldsymbol{D}$ are concatenated into a leader sample motion matrix, which denotes the root node of the HDb:

$$\boldsymbol{Y}^l = \left[\boldsymbol{y}_1^l, \boldsymbol{y}_2^l, \ldots, \boldsymbol{y}_m^l\right], \tag{7}$$

where m denotes the sum number of state vectors n in all demonstrations $\boldsymbol{D}$. Deeper levels are obtained by recursive k-means clustering, with $k = 2$, of each node's state vectors to produce two new nodes, as suggested in [2]. By dividing each node into just two clusters, the granularity of the encoded data changes gradually through the levels, which benefits the hierarchical search during the recognition step (detailed in Section II-B). In order to handle state vectors including both positions and quaternions, the distance metric used for k-means clustering had to be adapted[1]. While state vectors are used for determining clusters, additional data j_i and t_i is not, but is rather inherited. For each node v the mean of

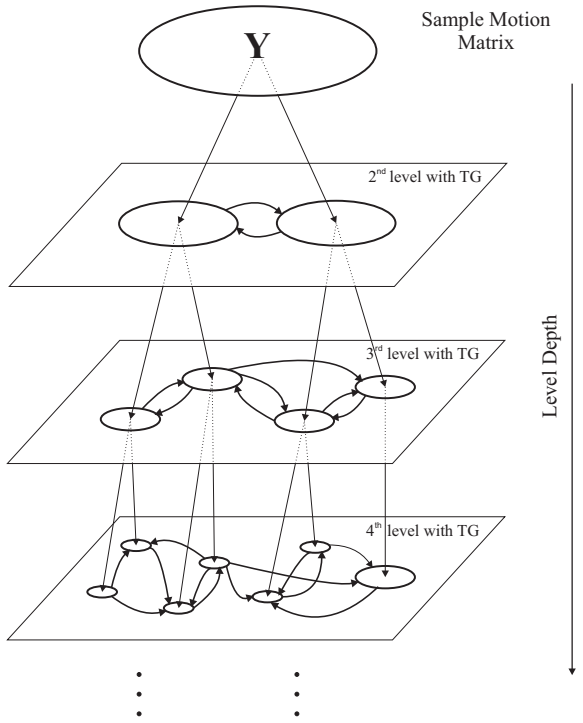

Fig. 3. A simplified representation of an example Hierarchical Database. A Sample Motion Matrix, which includes leader state vectors from all demonstrations in the set, represents the root node. Each level includes these state vectors clustered into nodes, where each level is built by clustering each node in the previous level. The probability of transition from one node into the other is represented with a Transition Graph (TG) at each level. The lineage of nodes is denoted by straight arrows. Each level in the database encodes demonstrated leader trajectories with increasing granularity, which enables a quick recognition of leader movements during execution.

associated state vectors $\overline{\boldsymbol{y}}_v^l$ is calculated and stored.[2]. The division of nodes continues recursively at each level until all nodes meet a stop criterion. This criterion is based on the variability of state vectors associated with a node, i.e. the distance of state vectors in the node v to the mean state vector of the node $\overline{\boldsymbol{y}}_v^l$. When the node stops dividing, it becomes a leaf node. To ensure all the data is represented on each level, leaf nodes are copied to all lower levels, i.e. each branch is extended to the last level.

All the demonstrated leader data is clustered in the process. The original state vector sequence is represented through directed weighted graphs. At each level of the hierarchical database, a Transition Graph (TG), which models all transitions between nodes at that level, is constructed (see Fig. 3). The graph's edge weights represent the probability of transitioning from one node to another, and are based on the number of transitions between nodes observed in the original demonstration data. The lineage of nodes, which is later used for recognition, is also stored. Further details on hierarchical

[1]The distance between two state vectors $\boldsymbol{y}_i^l = \left[\boldsymbol{p}_i^l, \boldsymbol{q}_i^l\right]$ is determined as a weighted sum of the Euclidean distance of the positional part of the state vector $d_p(\boldsymbol{p}_1^l, \boldsymbol{p}_2^l) = \sqrt{\boldsymbol{p}_1^l - \boldsymbol{p}_2^l}$ and the angle between the quaternions for the orientation part $d_q(\boldsymbol{q}_1^l, \boldsymbol{q}_2^l) = 2\arccos\left(q_{1w}^l q_{2w}^l + q_{1x}^l q_{2x}^l + q_{1y}^l q_{2y}^l + q_{1z}^l q_{2z}^l\right)$.

[2]In our case, the mean of N state vectors, $\overline{\boldsymbol{y}}^l = \left[\overline{\boldsymbol{p}}^l, \overline{\boldsymbol{q}}^l\right]$ represents the mean on the positional part $\overline{\boldsymbol{p}}^l$ and the approximate mean on the quaternion part, $\overline{\boldsymbol{q}}^l = \frac{1}{N}\sum_i^N \boldsymbol{q}_i^l$ The quaternion mean needs to be additionally normalized to ensure the result is a unit quaternion.

database construction are omitted and the reader is instead referred to [2], [3].

The HDb encodes just the leader movements. The follower state vectors are included by encoding each separate follower demonstration,

$$\boldsymbol{D}_j^f = \left[\boldsymbol{y}_1^f, \boldsymbol{y}_2^f, \ldots, \boldsymbol{y}_n^f\right], \tag{8}$$

as a Dynamic Movement Primitive (DMP)

$$\boldsymbol{D}_j^f \mapsto \left[\boldsymbol{w}_j^f, \boldsymbol{g}_j^f, \tau_j^f\right], \tag{9}$$

where $\boldsymbol{w}_j^f$ denotes the weights, $\boldsymbol{g}_j^f$, the goal, and τ_j^f, the time duration for each follower movement encoded as a DMP. If, as in the case presented in this paper, the follower trajectories include positions and quaternions, Cartesian space DMPs [35] are used. The trajectory representing the grasper opening angle is also encoded as a DMP. Details on DMPs are omitted and the reader is referred to [7], [8].

B. Recognition of Leader Movements

Hierarchical search through the primary database is used to recognize the leader's movement. Here, a sliding window approach is used. A sliding window is defined as the sequence of last w observed leader state vectors

$$\boldsymbol{Y}^W = \left\{\boldsymbol{y}_{b-w}^o, \ldots, \boldsymbol{y}_{b-1}^o, \boldsymbol{y}_b^o\right\}, \tag{10}$$

where o denotes *observed*, and b represents the complete number of observed leader state vectors from the start of the movement. Naturally, observed leader movements contain only leader's state vectors, $\boldsymbol{y}^l$, cf. (2). The size of the sliding window w is a compromise between the confidence of the result and recognition speed. Recognition is done by traversing through the levels of the HDb. Multiple steps are performed at each level:

1) Establish considered nodes v^c at current level. At the first level of the HDb used for recognition (usually level 3) all the nodes are denoted as considered nodes.
2) Build the matrix, $\boldsymbol{G}$, of considered nodes at the current level, where each row contains a permutation of considered nodes v^c and has the length of the sliding window w. Matrix $\boldsymbol{G}$ includes all possible permutations.
3) The recognition score, R, for each permutation set is calculated as

$$R = K_d \sum_{i=1}^{w} d\left(\boldsymbol{y}_i^o, \overline{\boldsymbol{y}}_v^l\right) + \sum_{i=1}^{w-1} \zeta\left(v_i^c, v_{i+1}^c\right), \tag{11}$$

where $d(\boldsymbol{y}_i^o, \overline{\boldsymbol{y}}_v^l)$ denotes distance[3] between the observed leader state vector $\boldsymbol{y}_i^o$ and the mean state vector $\overline{\boldsymbol{y}}_v^l$ corresponding to the considered node v_i^c. The distance weighting factor K_d is used to prioritize distance term in the overall score and is established empirically. The second term, $\zeta(v_i^c, v_{i+1}^c)$, denotes the transition probability between the considered nodes, which is derived from the TG at the current level.
4) Determine the nodes belonging to permutations with recognition score $R > R_c$, where the choice of cut-off threshold, R_c, is a compromise between the speed of recognition and the confidence of the result. Only the children of these nodes are considered as we move to the next level of the HDb. By taking advantage of the hierarchical structure of the database and only looking at a subset of nodes at each level, calculation time for the recognition step is reduced.

With the last level reached, the permutation of nodes with the highest recognition score is the recognized *leader sequence*. It represents the most probable sequence of nodes in the HDb w.r.t. the currently observed leader's movement during task execution. See Fig. 4 for a simple representation.

[3]In the case presented in this paper, the distance is determined as a weighted sum of the Euclidean distance of the positional part of the state vector $d_p(\boldsymbol{y}_i^o, \overline{\boldsymbol{y}}_v^l) = \sqrt{\boldsymbol{p}_i^o - \overline{\boldsymbol{p}}_v^l}$ and the angle between the quaternions for the orientation part $d_q(\boldsymbol{y}_i^o, \overline{\boldsymbol{y}}_v^l) = 2\arccos\left(q_{iw}^o \overline{q}_{vw}^l + q_{ix}^o \overline{q}_{vx}^l + q_{iy}^o \overline{q}_{vy}^l + q_{iz}^o \overline{q}_{vz}^l\right)$.

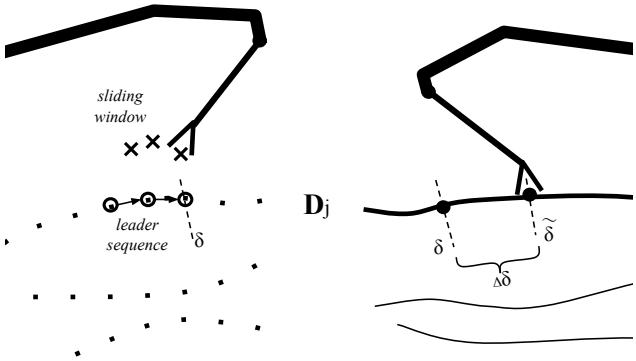

Fig. 4. A simple example of follower trajectory synthesis during the execution step. The leader robot, being controlled by the user, is depicted on the top left. Its last w state vectors, denoted by crosses, represents the current *sliding window*. HDb is used for recognition, and the most probable *leader sequence*, denoted with circles and arrows, is obtained. With it, the most probable demonstrated set D_j is determined. It also tells us how far along the set the current observation is (denoted with δ). To ensure smooth movements and velocity adaptation of the follower, DMPs are used. The difference between the desired δ and adapted $\tilde{\delta}$, denoted by $_\Delta\delta$, is used to determine the progression of the DMP, i.e. the velocity of the follower.

C. Synthesis of Follower Trajectories

By establishing the most probable *leader sequence*, we can determine which leader/follower demonstration set $\boldsymbol{D}_j$ is the most probable. This is done by looking at additional data of state vectors belonging to nodes in the determined *leader sequence*. Additional data includes information on set numbers j for each state vector.

We now know which demonstrated movement the follower should be executing, but not how far along the trajectory this movement should be. We denote this completion rate for the movement as $\delta = {}^t/_\tau$, where t is the current time and τ is the total duration of the movement. As the leader and follower movement share the time component, if we can estimate δ for the leader, we will also know how far along the movement the follower should be. Given the most probable *leader sequence*, we can establish that $\delta \approx {}^{\tilde{t}}/_{\tau_j}$, where τ_j is

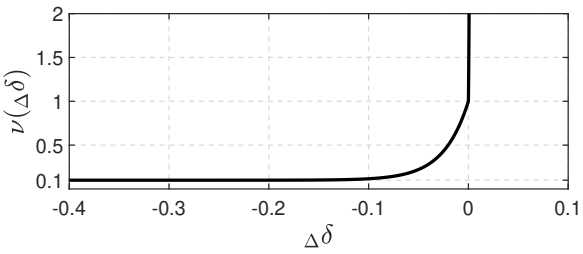

Fig. 5. An example of the speed scaling function $\nu(_\Delta\delta)$ w.r.t. the difference of path traveled $_\Delta\delta$. The further behind the actual follower is, the higher the velocity will be. When the difference becomes non-negative the value rises sharply, which in turn stops the follower. The minimum value of $\nu(_\Delta\delta)$ limits the maximum velocity of the follower. The function values used for this figure are throughout the evaluation: $\nu_{\text{min}} = 0.1$, $K_n = 40$, and $K_p = 1000$.

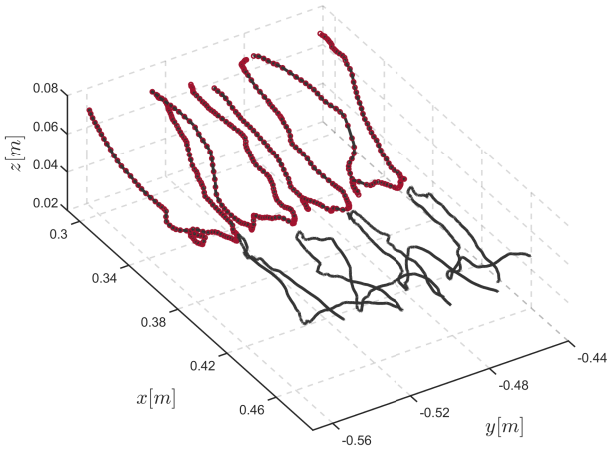

Fig. 6. Positional parts of the example movements. While the leader positions can be seen on the top left, the follower's are on the bottom right. Positional parts of the state vector means $\overline{\boldsymbol{y}}_v^l$ for each node at the last level are also seen. They are denoted with red circles.

the total duration of the recognized demonstrated movement. The current time component, $\widetilde{t}$, is estimated by looking at additional data corresponding to nodes in the current *leader sequence*; more specifically, at the state vector's time stamps t_i.

Before sending the *desired follower pose* to the follower robot, we wish to ensure two things: smooth follower movements and speed of execution relative to the speed of the controlled leader movement. If the user controlling the leader robot stops or changes velocity, the follower movement should do the same. Achieving smooth follower movements can be especially problematic when the recognized set $\boldsymbol{D}_j$ changes, as this could lead to an abrupt change in the desired follower pose. We tackle both of these problems through DMPs. Standard DMPs ensure smooth responses to perturbation, which in our case ensures smooth and continuous movement of the follower, as its pose is encoded as a DMP and integrated online. To tackle smooth changes to speeds of execution, DMPs with velocity adaptation [36], [37] are implemented. While the details on DMPs with velocity adaptation are omitted here, it should be noted that they differ from standard DMPs by multiplying the time-constant τ with a scaling function ν. This in turn changes the velocity of the DMP. While previous works on DMP velocity adaptation use different learning techniques to adapt the velocity profile offline, we need to adapt it online and ensure continuous and smooth synchronization to the leader's speed of execution.

To ensure smooth velocity adaptation based on the leader, the path traveled δ should not be used directly on the follower robot, as it can change abruptly. Instead it is used to control the progression of the follower DMP indirectly. While Nemec et. al [36] used learning techniques to define the scaling function ν, we define it based on the difference of path traveled,

$$_\Delta\delta = \widetilde{\delta} - \delta, \tag{12}$$

where $\widetilde{\delta}$ denotes how far along the full movement the actual follower is (see Fig. 4). The scaling function is defined as

$$\nu(_\Delta\delta) = \begin{cases} \exp\left(K_{p\,\Delta}\delta\right) & _\Delta\delta > 0 \\ \exp\left(K_{n\,\Delta}\delta + \log(1-\nu_{\text{min}})\right) + \nu_{\text{min}} & _\Delta\delta \leq 0, \end{cases} \tag{13}$$

where ν_{min} defines the smallest value for $\nu(_\Delta\delta)$. K_p and K_n define the slope for the exponential function, while the $_\Delta\delta$ is positive or negative, respectively. An example of $\nu(_\Delta\delta)$ can be seen if Fig. 5. If the user moves the leader far ahead, the recognized *leader sequence* will jump further along the demonstrated movement, which, in turn, will increase the value of δ. Instead of the follower jumping to that value, the difference in path traveled $_\Delta\delta$ will rise and in turn increase the velocity of the follower DMP. If the user slows down or stops, $_\Delta\delta$ will drop and the follower DMP will in turn slow down or stop. This ensures smooth velocity adaptation to the leader.

III. Evaluation

The evaluation setup consisted of a UR5 and UR5e robot arm (Universal Robots A/S) each with a EndoWrist Large Needle Driver surgical tools (Intuitive Surgical Inc.), and two Touch haptic devices (3D Systems), as can be seen in Fig. 1. Refer to [38] for more details about the surgical robot platform. The robot seen on the left is denoted as robot B, while the one on the right as robot A. Both arms could be controlled directly or by a human via haptic devices.

The proposed approach was evaluated with an ambidextrous peg transfer task, which is a part of the McGill Inanimate System for Training and Evaluation of Laparoscopic Skills (MISTELS) training program [39]. The test bed can be seen in the upper right corner of Fig. 1. Each ring is picked up by robot B, handed over to robot A, and positioned on the corresponding peg on the opposite side of the test bed.

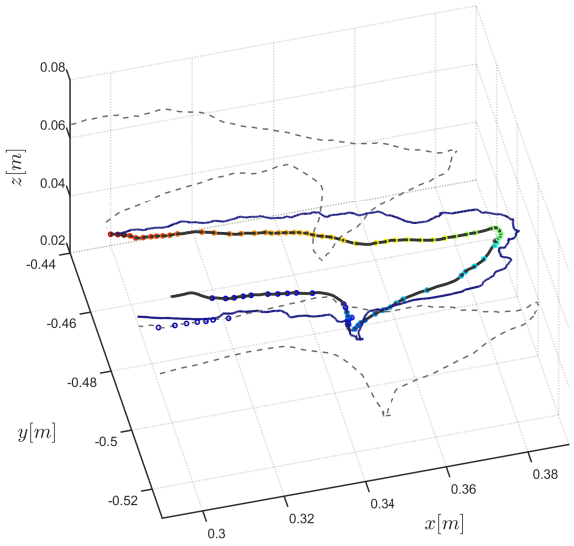

Fig. 7. Recognition during evaluation of cooperative task execution. The blue line denotes the position of the leader robot during evaluation of cooperative task execution. Demonstrated movements are denoted with dashed lines. The closest demonstrated movement to the current execution is presented with a solid black line. Overlaid on it as circles is the positional part of the mean state vector of the last nodes in each recognized *leader sequence*. Their color changes from dark blue to red w.r.t. the time progression of the cooperative task execution.

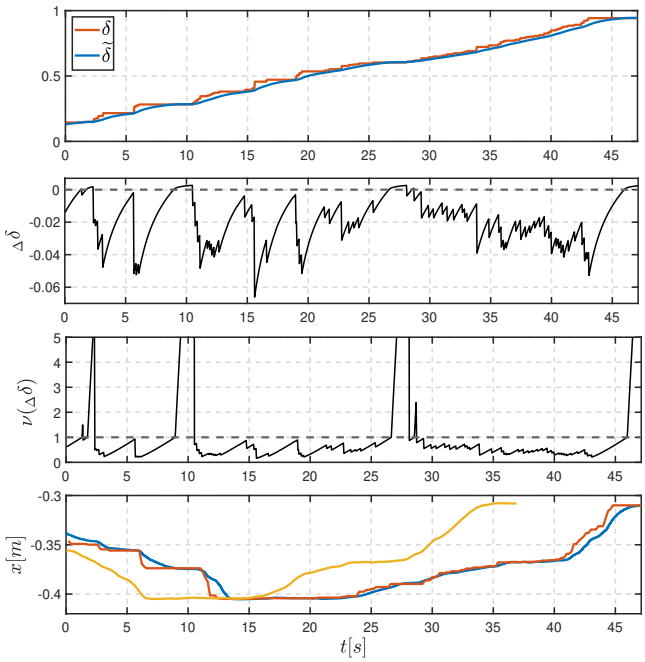

Fig. 8. Velocity adaptation. The top figure shows the progression of path traveled, of the current observation in red and of the actual follower in blue. The second plot from the top shows their difference $_{\Delta}\delta$ with the dashed line marking the value zero. The third plot from the top show the values of the scaling function $\nu(_{\Delta}\delta)$ over time, with dashed line marking value 1. The bottom plot is showing the x component of the follower's position. Red denotes the desired follower position based on observation, blue the actual adapted one, while yellow shows demonstrated values used for building the database. All figures show the same execution of a cooperative movement.

In the demonstration step, both arms were controlled via the haptic devices in order to capture four example dual-arm movements $\boldsymbol{D} = \{\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{D}_3, \boldsymbol{D}_4\}$. Each example movement transferred one of the four rings from the far-left position to the far-right position, as seen in Fig. 1. Positional parts of the example movements can be seen in Fig. 6.

Example sets of leader/follower movements were used to build the database, as explained in Section II-A. The first cluster of the HDb, i.e. the sample motion matrix $\mathbf{Y}^1$ included $m = 11135$ leader state vectors. When the HDb was built, it consisted of 14 levels with 482 nodes at the last level. The corresponding follower movements were encoded as set of DMPs. The same database was used throughout the evaluation.

During cooperative task execution, the user only controls the leader arm (robot B) via the haptic device, while the movement for follower arm (robot A) is synthesized by the presented approach. A sliding window of size $w = 3$ is used to determine the *leader sequence* and with it the most probable demonstration from $\{\mathbf{D_s}\}$, as described in Section II-B. Fig. 7 shows one example of cooperative task execution, where the nodes closest to current leader input are recognized.

In addition to replaying the recognized follower movement from the database, the follower should adapt to the leader's speed of execution. Fig. 8 shows the effects of using the speed scaling function (13) with DMPs. The top plot demonstrates that the follower can smoothly and continuously adapt to the observed path. It also holds its current position when it catches up to the observed one. As the difference of the path traveled, $_{\Delta}\delta$, shown in the second plot from the top, becomes positive, the scaling function, $\nu(_{\Delta}\delta)$, seen in the third plot from the top, rises sharply. This in turn increases the value of the DMP parameter τ and halts the current values of trajectories encoded in the DMPs. The bottom plot in Fig. 8 shows the x component of the follower's position. The red line denotes the values we would get without the speed scaling, i.e. follower positions based just on current observed leader position. Notice the sudden jumps in follower position compared to the blue line, which shows the actual progression of the follower robot position, and displays smooth and continuous adaptation to current observations. We can observe that the progression of the movement halts multiple times, when the user controlling the leader also halts. The yellow line denotes the values from the example movement in the database. Without velocity adaptation, the resulting follower trajectory would closely follow this line, the follower robot would overtake the leader movements, and the task would fail. While just one component is shown, the same observations could be made for the other position and orientation components.

If the leader changes task mid-execution, the follower should not just recognize this intent, but also react in a smooth and continuous manner. Fig. 9 shows an example of this. In the top plot, we can observe the proposed approach

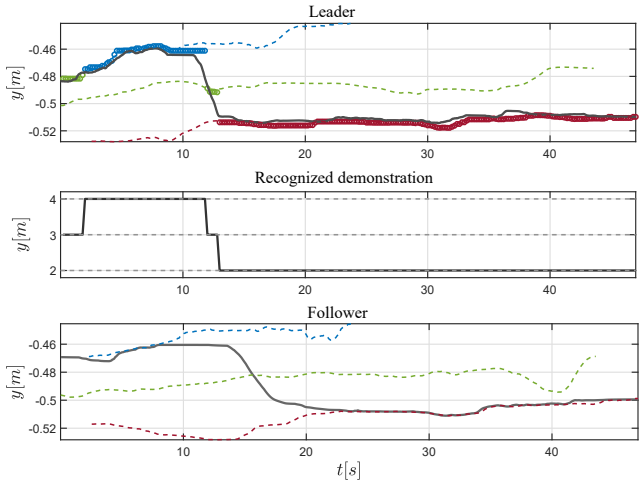

Fig. 9. Change in the recognized set. The top plot shows the y positional component related to the leader robot. The input values used for recognition in the sliding window are denoted with a solid black line. The values related to the most probable nodes are marked with circles, where the color indicates which demonstrated set those nodes belong to. The example values used for building the database are denoted with dashed lines, with the color indicating a set. Their time component is adapted in this figure for clarity. The middle plot shows which demonstrated set $\mathbf{D_j}$ was recognized as most probable. The bottom plot relates to the follower robot. Again, just the y component is shown. While the values of example sets are again denoted with dashed colored lines, the final synthesized values, used to control the follower robot, are presented with a solid black line.

where the most probable nodes are determined based on the user's leader robot inputs. The additional data in the nodes is then used to determine the most probable example set $\boldsymbol{D_j}$, shown in the middle plot. The synthesized trajectory, used to control the follower robot, reacts to the user's intent with smooth and continuous movements (see bottom plot). While only one positional component is shown, the same could be observed for all other positional and orientational components.

IV. Discussion

While we have demonstrated the adaptability of our synthesis scheme for follower trajectories to both changes in the recognized leader motion and changes in the speed of execution, further discussion of the practical implications of this are needed.

We currently rely on only the surgical tools' position for recognizing the current leader intent. This is a problem for a task such as the peg transfer task, where the exact trajectory toward the ring is not important for the outcome of the task (ring moved to another peg), unless obstacles are in the way. To improve this, our approach could be extended by including information about the environment. For example, in the presented peg transfer task, the ring and peg positions could also be included in the state vectors. This would require computer-vision-based detection and localization which we have not yet included in the system, however.

Note that, although our evaluation has shown adaptation over a limited set of examples, the approach is not strictly limited to encoded demonstrations. Instead, statistical generalization could be used to interpolate between encoded demonstrations, thus detecting leader motions not in the original set of demonstrations. As follower motions are encoded as DMPs, generalization could then be used to synthesize appropriate corresponding follower trajectories. Additional state variable could be used for generalization, e.g. generalization w.r.t. peg positions.

The surgical tools have a good deal of error in state estimate due to elasticity and friction in the steel cables used for actuating the joints. The is especially noticeable in end-effector orientation and grasper opening angle of the tools. During execution, this can potentially cause erroneous recognition of the leader's motion. This can be alleviated by weighting the sum of positional and orientational components in the distance metric.[3]

Finally, we remark that the task showcased in our evaluation was chosen to clearly illustrate the properties of our proposed approach. However, our method could be applied to more complex collaborative scenarios, and is not constrained to only bi-manual tasks. In future studies, we plan to evaluate our approach on a three- or four-armed surgical system, where the operator uses two arms to perform, e.g. a suturing task, while a third follower arm is autonomously controlled and assists with, e.g. holding tissue or endoscope arm movement. We also plan to evaluate the possibility of building the database from multiple executions and thus encode the variability of demonstrations.

V. Conclusions

We have presented an approach towards partial automation of robotic surgical tasks. By encoding a set of example bimanual tasks in a database, the user can execute a cooperative task while only controlling a single robot arm. The movements of a second autonomous follower arm are then matched to the example in the database that most closely resembles the operator's estimated intent. Compared to previous works, the recognition step was improved and a novel approach for synthesizing follower trajectories synchronized to leader's speed of execution was presented. The evaluation, performed in a non-clinical surgical environment, showed that the recognition based on leader's pose is done fast enough for real-time task execution, and that the follower adapts to the leader's intent and speed throughout the execution in a smooth and continuous manner. Future work will include evaluation of a database with more diverse and demanding tasks (e.g. suturing) in an environment that more closely resembles a clinical surgical scenario.

Acknowledgment

We would like to thank Michael Kjær Schmidt and Nicolai Iversen, both with the Mærsk McKinney Møller Institute, University of Southern Denmark, for their help with our experimental setup.

REFERENCES

[1] R. H. Taylor, A. Menciassi, G. Fichtinger, P. Fiorini, and P. Dario, "Medical Robotics and Computer-Integrated Surgery," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., 2nd ed., Springer International Publishing, 2016, pp. 1657–1684.

[2] M. Deniša and A. Ude, "Synthesis of new dynamic movement primitives through search in a hierarchical database of example movements," *Int J Adv Robot Syst.*, vol. 12, no. 137, 2015.

[3] M. Deniša, T. Petrič, T. Asfour, and A. Ude, "Synthesizing compliant reaching movements by searching a database of example trajectories," in *In Proc. IEEE Int Conf on Humanoid Robots*, 2013, pp. 540–543.

[4] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, 1999.

[5] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends Cogn. Sci.*, vol. 6, no. 11, pp. 481–487, 2002.

[6] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robot. Auton. Syst.*, vol. 47, no. 2, pp. 109–116, 2004.

[7] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control–a unifying view," *Prog. Brain Res.*, vol. 165, pp. 425–445, 2007.

[8] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013.

[9] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 800–815, 2010.

[10] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robot. Auton. Syst.*, vol. 60, no. 10, pp. 1327–1339, 2012.

[11] H. Ben Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human-robot cooperation tasks," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 2831–2837.

[12] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robot. Automat. Mag*, vol. 17, no. 2, pp. 44–54, 2010.

[13] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, 2011.

[14] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2013, pp. 2616–2624.

[15] T. Inamura, I. Toshima, H. Tanie, and Y. Nakamura, "Embodied symbol emergence based on mimesis theory," *Int. J. Robot. Res.*, vol. 23, no. 4-5, pp. 363–377, 2004.

[16] H. B. Amor, D. Vogt, M. Ewerton, E. Berger, B. Jung, and J. Peters, "Learning responsive robot behavior by imitation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2013, pp. 3257–3264.

[17] D. Lee, C. Ott, and Y. Nakamura, "Mimetic communication model with compliant physical contact in human—humanoid interaction," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1684–1704, 2010.

[18] H. B. Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human-robot cooperation tasks," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 2831–2837.

[19] M. Deniša, B. Nemec, and A. Ude, "Cooperative movements through hierarchical database search," in *Proc. IEEE Int. Conf. Advanced Robotics (ICAR)*, 2017, pp. 40–46.

[20] K. Yamane, Y. Yamaguchi, and Y. Nakamura, "Human motion database with a binary tree and node transition graphs," *Autonomous Robots*, vol. 30, no. 1, 2011.

[21] K. Yamane, M. Revfi, and T. Asfour, "Synthesizing object receiving motions of humanoid robots with human motion database," in *Proc. IEEE Int Conf on Robotics and Automation (ICRA)*, 2013, pp. 1629–1636.

[22] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X. Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010, pp. 2074–2081.

[23] N. Padoy and G. D. Hager, "Human-machine collaborative surgery using learned models," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 5285–5292.

[24] A. Knoll, H. Mayer, C. Staub, and R. Bauernschmitt, "Selective automation and skill transfer in medical robotics: A demonstration on surgical knot-tying," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 8, no. 4, pp. 384–397, 2012.

[25] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, "A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2013, pp. 4111–4117.

[26] A. Murali, S. Sen, B. Kehoe, A. Garg, S. McFarland, S. Patil, W. D. Boyd, S. Lim, P. Abbeel, and K. Goldberg, "Learning by observation for surgical subtasks: Multilateral cutting of 3d viscoelastic and 2d orthotropic tissue phantoms," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 1202–1209.

[27] M. Power, H. Rafii-Tari, C. Bergeles, V. Vitiello, and G.-Z. Yang, "A cooperative control framework for haptic guidance of bimanual surgical tasks based on learning from demonstration," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 5330–5337.

[28] P. Berthet-Rayne, M. Power, H. King, and G.-Z. Yang, "Hubot: A three state human-robot collaborative framework for bimanual surgical tasks based on learned models," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 715–722.

[29] T. Osa, N. Sugita, and M. Mitsuishi, "Online trajectory planning and force control for automation of surgical tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 675–691, Apr. 2018.

[30] M. Ginesi, D. Meli, A. Roberti, N. Sansonetto, and P. Fiorini, "Autonomous task planning and situation awareness in robotic surgery," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2020, pp. 3144–3150.

[31] K. L. Schwaner, I. Iturrate, J. K. H. Andersen, P. T. Jensen, and T. R. Savarimuthu, "Autonomous bi-manual surgical suturing based on skills learned from demonstration," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, to be published, Prague, Czech Republic: IEEE, Sep. 2021.

[32] G. De Rossi, M. Minelli, S. Roin, F. Falezza, A. Sozzi, F. Ferraguti, F. Setti, M. Bonfè, C. Secchi, and R. Muradore, "A first evaluation of a multi-modal learning system to control surgical assistant robots via action segmentation," *IEEE Transactions on Medical Robotics and Bionics*, vol. 3, no. 3, pp. 714–724, 2021.

[33] M. Hwang, D. Seita, B. Thananjeyan, J. Ichnowski, S. Paradis, D. Fer, T. Low, and K. Goldberg, "Applying depth-sensing to automated surgical manipulation with a da vinci robot," in *2020 International Symposium on Medical Robotics (ISMR)*, IEEE, Nov. 2020.

[34] G.-Z. Yang, J. Cambias, K. Cleary, E. Daimler, J. Drake, P. E. Dupont, N. Hata, P. Kazanzides, S. Martel, R. V. Patel, *et al.*, "Medical robotics—regulatory, ethical, and legal considerations for increasing levels of autonomy," *Science Robotics*, vol. 2, no. 4, p. 8638, 2017.

[35] A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 2997–3004.

[36] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *Proc. IEEE Int. Conf. Humanoid Robots (Humanoids)*, 2013.

[37] R. Vuga, B. Nemec, and A. Ude, "Speed adaptation for self-improvement of skills learned from user demonstrations," *Robotica*, vol. 34, no. 12, pp. 2806–2822, 2016.

[38] K. L. Schwaner, I. Iturrate, J. K. H. Andersen, C. R. Dam, P. T. Jensen, and T. R. Savarimuthu, "Mops: A modular and open platform for surgical robotics research," in *2021 International Symposium on Medical Robotics (ISMR)*, to be published, Atlanta, GA, USA: IEEE, Nov. 2021.

[39] A. M. Derossis, G. M. Fried, M. Abrahamowicz, H. H. Sigman, J. S. Barkun, and J. L. Meakins, "Development of a model for training and evaluation of laparoscopic skills," *The American Journal of Surgery*, vol. 175, no. 6, pp. 482–487, 1998.