2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

# Human-Machine Skill Transfer Extended by a Scaffolding Framework

H. Mayer, D. Burschka and A. Knoll
Robotics and Embedded Systems
Technical University Munich
{mayerh|burschka|knoll}@in.tum.de

E.U. Braun, R. Lange and R. Bauernschmitt
Department of Cardiovascular Surgery
German Heart Center Munich
{brauneva|lange|bauernschmitt}@dhm.mhn.de

*Abstract*— **The term scaffolding, with respect to human education, was first coined in the 1970ies, although the basic concept originates back to the 1930ies. The main idea is to formalize the superior knowledge of a teacher in a certain way to generate support for a trainee. In practice, this concept can be implemented as concrete as a cloze, which assists pupils in learning a foreign language, or it might be as abstract as a social environment, which facilitates learning of specific tasks. This paper introduces a novel approach towards robotic learning by means of such a scaffolding framework. In this case, the scaffolding is constituted by abstract patterns, which facilitate the structuring and segmentation of information during "Learning by Demonstration". The methodology was applied to a real-world scenario of robot-assisted surgery.**

*Index Terms*— **learning by demonstration, scaffolding, situated learning**

## I. Introduction

The educational concept of scaffolding is closely related to the paradigm of situated learning. The latter refers to the fact that successful learning of new tasks can be effected, only if teacher and trainee are aware of a common environment, in which learning takes place. This particularly applies to the transfer of rather technical skills, as it is the case for surgical knot-tying, which will serve as application example for the presented methodology. One common implementation of situated learning, which has also been adopted for human-machine skill transfer, is learning by demonstration. By utilizing this concept, the teacher shows the execution of a technical task directly within the corresponding environment. In doing so, the teacher has to pay special attention to the capabilities of the trainee. I.e. learning of the task must not include knowledge, which cannot be perceived by the trainee. For example, teaching a sorting task to a blind person cannot be successful, if the teacher (unconsciously) utilizes color information of the items, going to be sorted. Therefore, the introduced concepts play also an important role in the education of handicapped persons [1], [2]. Transferred to robotics, the awareness of the capabilities of the robot plays an important role, since "learning by demonstration" was originally intended for users, which are no experts in programming robots. Compared to human senses, most robots are still underequipped with sensors today. Therefore, a human teacher, which is not aware of the robot's technology, might be confused by these shortcoming: Tasks which are easily performed by humans, like rotating an egg with the fingers of a single hand, can hardly be realized by robotic systems. On the other hand, robots undoubtly exceed humans in certain abilities (e.g. accuracy and speed). This potential will be wasted, if learning is reduced to the lowest common denominator of the capabilities of both humans and robots. This applies to most applications of learning by demonstration, since the complexity of the task has to be restricted to both the limited accuracy of the human teacher and the limited sensori-motor abilities of the robot. Therefore, including the non-overlapping parts into learning procedures can significantly support the learning process and will develop the treatment of complex tasks.

Inclusion of this information can be achieved by means of a scaffolding framework. With respect to learning, scaffolding refers to an assistance, which is generated from the superior knowledge of the teacher. On the one hand, this framework should guarantee a successful completion of the task, on the other hand, it should be generally enough to allow for new experiences of the trainee. Being a comprehensive term, a scaffolding framework can be as simple as a cloze, or it can comprise the whole social environment where learning takes place. Within the context of this work, the scaffolding framework is restricted to abstract patterns, which are designed by the user to provide templates of skill demonstrations. All succeeding demonstrations are validated against the template of the corresponding task. By means of the abstract patterns, which are used to compose the task template, the demonstrations are segmented into meaningful primitives. This leads to a formalization and reduction of the unstructured data. I.e. the coordinate-based data points of the user demonstrations are transferred to a generic description of the skill, which can be applied to new environments.

The concept of scaffolding traces back to the work of L. Vygotsky in the 1930s [3]. The expression scaffolding itself has first been coined by Wood et al. [4] and refers to the teacher giving assistance on certain aspects of a task, which are beyond the capability of the trainee. However, both authors exclusively focus on human learning. Robotic learning within a scaffolding framework has already been adopted by some research groups, but within completely different contexts. While some authors refer to the social environment, where robotic learning takes place [5], other groups have used scaffolding as a filtering method for sensor data, which was acquired by mobile robots [6]. The sensor readings have been fused into state vectors and scaffolding was used to weight the vectors according to the encountered situation, where the input from one sensor might be more important than the one of others (e.g. dominance of visual input over range sensors).

## II. Materials and Methods

A critical milestone in the development of new methodologies for robotic systems has always been their application in real-world scenarios. While conclusions drawn from simulations are comparably smooth, many awkward and unexpected details first occur, once the procedures are carried out on real robots posing typical challenges like calibration errors and limited dynamics. Therefore, the findings of this research project have been assessed within a realistic scenario of robotic heart surgery. Minimally invasive knot-tying has been chosen as a benchmark task, because it provides an extent of complexity, which requires new strategies for structuring and transferring information. After testing the algorithms in a simulation environment, all experiments have been repeated with the real-world surgical system introduced below.

### A. Robotic System

Fig. 1. **Hardware Setup:** Ceiling mounted robots with surgical instruments

The experiments have been conducted with a system for robot-assisted minimally invasive surgery, which was developed by the authors and already introduced to the research community in [7]. Therefore, the description of the hardware is restricted to an extent necessary for the understanding of the further parts of this paper. The slave manipulator of the system consists of four robots (Mitsubishi MELFA 6SL™, *Mitsubishi Electric Corp.*), which are mounted on a gantry on the ceiling (cf. fig. 1). The robots are equipped with minimally invasive instruments, which are originally deployed with the daVinci™surgical system [8] (*Intuitive Surgical, Inc.*). The robot is concatenated with the surgical instrument by a magnetic coupling, which prevents the instrument from damages in case of a severe collision. The instruments are powered by small servo motors, which are integrated into the coupling mechanism. Optionally, one of the robots can be equipped with a stereo camera instead of an instrument. After several tests of the ergonomics, a setup depicted in fig. 2 has been chosen as user interface. It consists of an aluminum frame, which can be quickly adapted to new geometries. The user's place is located in front of the main in-/output devices, two PHANToM™haptic displays. Their controller boxes are stored at the base of the frame.

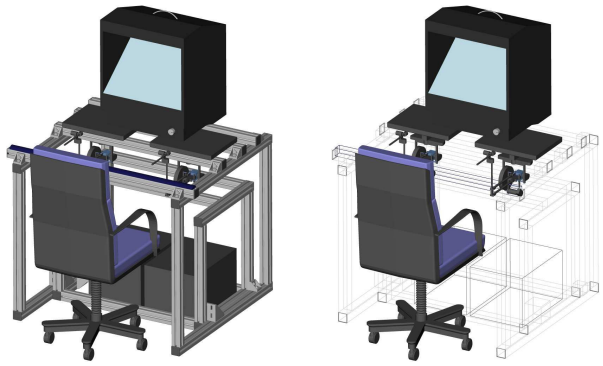


Fig. 2. **Master console with PHANToM devices and 3D screen**

The PHANToM's themselves are assembled upside down. This arranges for less constricted flexibility of the stylus pen. The 3D display is placed on top of the frame in a way not reducing the working space of the PHANToM's. As an additional input modality, foot switches are placed at the footwell of the console. One is dedicated as emergency exit while the function of the others can be arbitrarily engaged by software. In addition, forces occurring at the instruments are measured by strain gauge sensors and fed back by means of the haptic devices.

### B. Simulation environment

For offline evaluation of trajectories, a simulation environment of the system has been developed (cf. fig. 3). The GUI comprises an interface to a 3D model of the scene, which can be manipulated in realtime. For each object in the scene, a context menu can be displayed (on the left side) by clicking on the corresponding model. This provides a possibility to adjust the parameters of the underlying object. For example

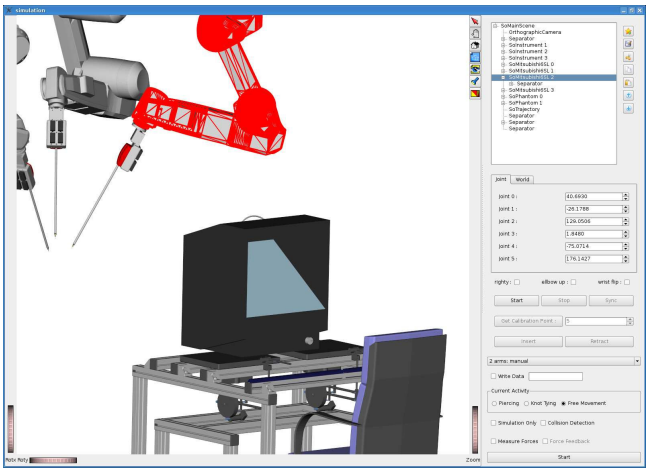


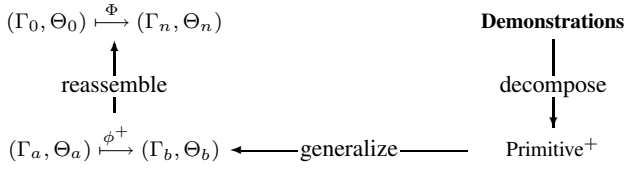
Fig. 3. **simulation environment:** Interventions can be assessed offline

the joint angles of the robots can be altered this way. In addition, the simulation environment also incorporates a key framing module and all trajectories constructed in the simulation can be checked by a collision detection.

### C. Skill transfer framework

Like the approaches of other authors (e.g. [10]), the presented skill transfer framework is based on the well-known technological paradigm of splitting up complex tasks into smaller parts. Transferred to robotic learning this paradigm means to assemble task-specific skills from a set of sensorimotor primitives [11]. The embodiment of certain primitives can be quite complex, if information from different sensors has to be included: e.g. in order to provide a primitive for controlling a multifingered hand [12] or for the force-controlled insertion of a peg into a hole [13]. For this examples, the primitives had to be pre-programmed by an experienced user. This restricts the composition of new skills to some combinatorial variations of the given primitives [14]. Therefore, later approaches tried to derive primitives from user demonstrations in order to provide more flexibility. In its original form, this paradigm implies that primitives have to be detected in user demonstrations without any contextual information, only based on cues extracted from unsupervised data mining like cluster analysis (e.g. k-means algorithm), inflexion points of the trajectory or occurring contacts (if those are detected by force sensors).

Some authors (e.g. [15]) already mentioned that a direct, unsupervised extraction of primitives from user demonstrations is difficult and often not feasible if the underlying trajectory is complex as for surgical knot-tying. Therefore, the pedagogical method of scaffolding is employed to support the derivation of primitives: the system is provided with patterns (descriptions of tasks), which specify the features to look for in user demonstrations. To give an informal example, such a pattern can be like: look for a linear motion between the closing of the gripper and a position where force is applied. At this point neither coordinates of this linear motion nor its speed are specified. These parameters will be extracted from user demonstrations if the corresponding pattern is detected within. This methodology can be summarized with the following diagram:




The **scaffolding framework** is constituted by a description of the task $\Phi$ and its decomposition into smaller parts, so-called tasklets $\phi$ (the plus sign in $\phi^+$ indicates that a task consists of at least one tasklet). Given this template the demonstration of the user is decomposed into a sequence of primitives, which corresponds to the decomposition of the task into tasklets as it is provided by the scaffolding framework. While a skill (the user's demonstration) is the instantiation of a certain task, primitives are instantiations of tasklets. Like every task, a tasklet is a transformation between pre- and post-conditions of an environment. Here, an environment is a set of objects $\Gamma$ and a set of properties $\Theta$ of these objects. The properties are defined quite generally and can refer to geometric quantities (like rigid transforms or distances) as well as forces or torques exerted to the object. A task is a transformation of an environment at $t_0$ into an altered environment at $t_n$, which will be expressed by $(\Gamma_0, \Theta_0) \overset{\Phi}{\longmapsto} (\Gamma_n, \Theta_n)$. The transition $\overset{\Phi}{\longmapsto}$ constitutes the **task** $\Phi$. In many cases $\Gamma_0$ will be equal to $\Gamma_n$ as long as the set of objects remains unchanged. The set will be changed, for example, if objects are destroyed or new objects are generated out of existing ones. For clarification refer to the following description of the well known peg-in-a-hole task: The environment is defined by the set of objects $\Gamma_{ph} = \{\text{peg, hole}\}$ with $\Theta_0$ comprising the rigid transform $T_{ph}$, which denotes the position of the peg relative to the hole. Therefore, the peg-in-a-hole task can be expressed as $(\Gamma_{ph}, \{T_{ph}\}) \overset{\Phi}{\longmapsto} (\Gamma_{ph}, \{I\})$, i.e. the goal is to align the peg with the hole until their centers are matched, and thus, the transformation between peg and hole finally constitutes a unit matrix $I$. Task definitions provide the system with abstract information about a task, which is going to be demonstrated by the user. Providing this kind of hints is the very essence of **scaffolding** as it is used in our skill-transfer architecture.

Given the definitions introduced above, a tasklet will be denoted by a corresponding transition: $(\Gamma_a, \Theta_a) \overset{\phi}{\longmapsto} (\Gamma_b, \Theta_b)$, where $\overset{\phi}{\longmapsto}$ is the transition of tasklet $\phi$. Therefore, a task is a set of tasklets, which met the following requirements:

$$\exists\, t_b \leq tn : (\Gamma_0, \Theta_0) \overset{\phi}{\longmapsto} (\Gamma_b, \Theta_b) \tag{1}$$

$$\exists\, t_a \geq t0 : (\Gamma_a, \Theta_a) \overset{\phi}{\longmapsto} (\Gamma_n, \Theta_n) \tag{2}$$

$$\left\{\overset{\Phi}{\longmapsto}\right\} \in \left\{\overset{\phi}{\longmapsto}\right\}^+ \tag{3}$$

The first requirement determines the pre-condition of task $\Phi$ being also a pre-condition of at least one tasklet (the same accounts to the post-condition in the second requirement). The last statement refers to the transitive hull of the tasklet transitions: the transition of task $\Phi$ has to be included in the transitive hull in order to let these tasklets be a valid dissection of the task. For a complete description of the task, the transitions of the tasklets have to be ordered appropriately. Therefore, we need to extend our definition of a task by three different relations. Those are irreflexive, transitive relations defined on the set of tasklets $\phi^0 ... \phi^m$. The most important is the precedence relation "$<$", i.e. $\phi_0 < \phi_1$ self-evidently means tasklet $\phi_0$ has to be completed before tasklet $\phi_1$ starts. In addition, we need a synchronization relation "$=$". Accordingly, $\phi_0 = \phi_1$ indicates that the tasklets have to be carried out at exactly the same starting time, and also implies that execution times of both tasklets are equal. For integrity reasons we also define an exclusion relation "$\neq$", which prevents tasklets from being executed at the same time. An application example for this type of relation would be a situation where two manipulators cannot move to the same place at the same time, but the corresponding tasklets

are not ordered by any precedence. Putting it all together we get the following **extended definition of a task**:

$$\Phi = \{\phi^*, <, =, \neq\} \tag{4}$$

While the transition of a task $\Phi$ is determined in quite an abstract manner (transform environment $(\Gamma_0, \Theta_0)$ into environment $(\Gamma_n, \Theta_n)$), we will define four concrete types of tasklets: linear motion, 2D motion, force controlled motion and synchronized motion. Those four will suffice to construct our application example of surgical knot tying. A **linear motion** is constituted by the following tasklet:

$$(\Gamma_a, \{T_a\}) \overset{\phi_{\text{lin}}}{\longmapsto} (\Gamma_b, \{T_b\}) \tag{5}$$

$\Gamma_a$ contains at least the manipulator, which is going to carry out the movement. In the pre-condition the manipulator is placed at posture $T_a$ and finally should reach $T_b$ in the post-condition. Both postures are interconnected by a linear movement regarding the translational part. Rotations are calculated by a spherical linear interpolation based on quaternions [9]. A more general version of this tasklet is the **2D motion**:

$$(\Gamma_a, \{T_a\}) \overset{\phi_{\text{2D}}}{\longmapsto} (\Gamma_b, \{T_b\}) \tag{6}$$

where both postures are interpolated by an arbitrary 2D spline, which interconnects the 3D coordinates of $T_b$ and $T_b$. Again, rotations are handled by spherical linear interpolation. Note that this definition poses no further restrictions on the spline, except embedding into a plane. The next tasklet, the **force restricted motion** is basically a linear motion:

$$(\Gamma_a, \{T_a, F_a\}) \overset{\phi_{\text{F}}}{\longmapsto} (\Gamma_b, \{T_b, F_b\}) \tag{7}$$

The difference is that the motion will be stopped before $T_b$ is reached, if a certain force $F_b$ is exceeded. In this case, the rest of the rotational motion will be carried out in place. The last tasklet we want to introduce here is the **synchronized motion**. This one is special, because, as a pre-condition, it requires another tasklet $\phi_s$ to exist, which is not a synchronized motion tasklet:

$$(\Gamma_a, \{T_a, \phi_s\}) \overset{\phi_{\text{syn}}}{\longmapsto} (\Gamma, \Theta) \tag{8}$$

The post-condition is arbitrary since the purpose of this tasklet is not directed towards a certain state of the environment, but it is correlated with the trajectory of another tasklet $\phi_s$. This correlation can be realized by an affine bijection.

Different combinations of these tasklets can be used to form a task. In the upper left corner of fig. 4 the task of surgical knot-tying is depicted (performed by three arms $\Rightarrow$ three tracks). Each tasklet is displayed by a colored rectangle. "2D" stands for a 2D primitive, "lin" for a linear motion, "F" for a force controlled primitive and "Sync" for a synchronized motion. The vertical red lines denote synchronization points. After all, it is possible to construct a broad variety of different tasks out of only four tasklets as they are defined above.

### D. Feature Extraction

In order to extract the relevant features from user demonstrations, the corresponding trajectories are pre-processed by an unsupervised event detection algorithm, which identifies certain states of the environment as they are defined in the $\Theta$-part of the tasklets. Those states are used to match tasklets against the actual demonstration in order to derive corresponding primitives. The extraction algorithm can be seen as a finite state machine, whose states are defined by the tasklets. This finite automaton processes the user input (demonstration) and checks whether it was an instance of the task definition or not. After each state change (each recognition of a tasklet), the automaton separates a primitive from the input and produces a corresponding output. For example, if an instantiation of a force controlled movement tasklet was detected in the input, the algorithm will extract the maximum force from the demonstration. This information is saved in order to instantiate the tasklets in a new environment. I.e. once all primitives are derived from the demonstration, the relevant information is stored in the knowledge base of the system. The structure of the knowledge base and the procedure of inserting information is depicted in fig. 4.

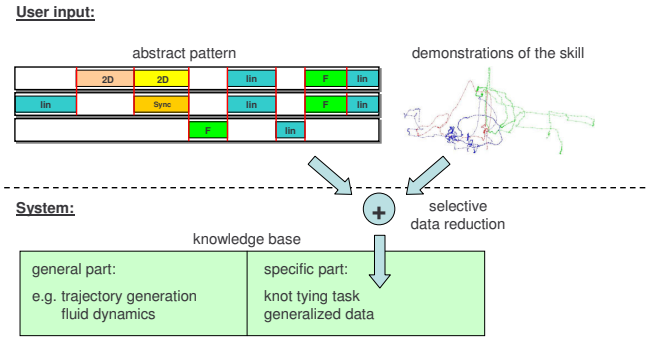



Fig. 4. **Knowledge base of the system**

The knowledge base consists of a general and a specific part. The general part contains information about the execution of primitives, which can be utilized in every task. For example, the general knowledge base comprises a trajectory generator in order to implement the straight line movements of linear motion primitives. Another example is a fluid simulation for the instantiation of 2D primitives [7]. This part of the knowledge base is hard-wired and will not be changed by user interactions. The only situation, which requires an alteration of the general knowledge base is the extension of the system by new tasklets. This has to be effected by adjusting the source code of the system. During normal operation, the user can only change the task-specific part of the knowledge base. This contains the definition of the task (in our case surgical knot-tying with three robotic arms) and extracted information from the demonstrations, which is necessary to actually instantiate the task in real world environments. The corresponding task-dependent information is extracted by selective data reduction of the primitives. There is a special method of data extraction for each type of tasklet:

1) **2D movement**: After the pre-processing mentioned above, the points constituting the primitives are interpolation points of a spline representation of the trajectory. The central part of feature extraction for 2D movements is an optimization algorithm, which calculates an optimal plane for a given set of 3D points (i.e. the primitive) Optimality is defined by means of the minimum least squares method. The regression problem is stated as follows:

$$^{2D}z_i = ax_i + by_i + c; d_i = z_i - {}^{2D}z_i; Min\left(\sum_{i=0}^{n} d_i^2\right) \quad (9)$$

where $(x_i, y_i, z_i)$ is the $i$th point of the corresponding primitive. The minimization is actually performed by linear regression. If the mean squared error exceeds a certain threshold (i.e. the points does not fit well to a 2D plane) the whole demonstration will be rejected for not being an instance of the predefined task.

2) **Linear movement**: If a linear motion primitive was detected in the demonstration, it suffices to store the start and end point of the underlying trajectory. All other points in between can be omitted.

3) **Force controlled movement**: As for the linear motion, the start and end point of the primitive is stored in the task-specific knowledge base. In addition, the maximum force vector during the movement is extracted. This will be used later to control the execution of the instantiation of this tasklet.

4) **Synchronized movement**: The synchronized movement tasklet can only exist in connection with a 2D tasklet. The points of the corresponding 2D primitive are mapped onto the points of the synchronized movement primitive. We have applied an affine transformation to describe this mapping:

$$\begin{pmatrix} x_1 \cdots x_n \\ y_1 \cdots y_n \\ z_1 \cdots z_n \\ 1 \cdots 1 \end{pmatrix} = A \begin{pmatrix} {}^{2D}x_1 \cdots {}^{2D}x_n \\ {}^{2D}y_1 \cdots {}^{2D}y_n \\ {}^{2D}z_1 \cdots {}^{2D}z_n \\ 1 \cdots 1 \end{pmatrix} \quad (10)$$

In order to calculate transformation $A$, each point in the trajectory of the referenced 2D primitive has to correspond to a point in the synchronized movement primitive. Therefore, the number of points $n$ used for this procedure has to match for both primitives. Finding the affine transformation is formulated as the following singular value decomposition problem: $C = \frac{1}{n}\sum_{i=1}^{n} \left[\ \overrightarrow{p_i} - \overline{p}\ \right] \left[\ \overrightarrow{{}^{2D}p_i} - \overline{{}^{2D}p}\ \right]^T$

$$\xrightarrow{\text{svd}} \quad C = USV^T \quad (11)$$

$$\text{where } \overline{p} = \frac{1}{n}\sum_{i=1}^{n} \overrightarrow{p_i};\ \ \overline{{}^{2D}p} = \frac{1}{n}\sum_{i=1}^{n} \overrightarrow{{}^{2D}p_i} \quad (12)$$

$\overrightarrow{p_i}$ is the $i$-th point of the currently processed primitive, while $\overrightarrow{{}^{2D}p_i}$ is the $i$-th point of the 2D primitive the former is synchronized with. Note that both primitives are stored as splines, and therefore, both can be resampled with an equal number of $n$ points. The diagonal of $S$ contains the eigenvalues $s_1, s_2$ of matrix $C^TC$. $U$ and $V$ contain the eigenvectors of $CC^T$ and $C^TC$, respectively. By means of these results, we can determine the scaling factor $f$, the 2D rotation matrix $R$ and the translation vector $\overrightarrow{t}$, which can be used to map $^{2D}p_i$ onto $p_i$:

$$f = \frac{s_1 + s_2}{\sigma_P} \ \ \textit{where} \ \ \sigma_P = \frac{1}{n}\sum_{j=1}^{n}\left(\overrightarrow{{}^{2D}p_i} - \overline{{}^{2D}p}\right) \quad (13)$$

$$R = f \cdot UV \quad (14)$$

$$\overrightarrow{t} = \overline{p_i} - f \cdot R\overline{{}^{2D}p_i} \quad (15)$$

## III. Results

As mentioned in the introduction, the presented skill transfer framework was employed to learn the performance of a surgical knot. Although some groups already tried to transfer knot-tying theory into applications of robotic learning [16], there has been only little research on the particular task of automating a surgical knot. Hynes et al. [17] proposed a robotic setup for knot-tying, but they did not evaluate the task under realistic circumstances, yet (i.e. small-scale knot performed under the restrictions of trocar kinematics and with original suture material). There is also some work available on analyzing the knot-tying task itself, but they are rather focused on surgical evaluation than on automation [18]. In addition, a generalized procedure for surgical assistance by using surgical fixtures has been proposed by a research group of the *Johns Hopkins Haptics Lab* [19]. This method could be used for further automation of surgical knot-tying (e.g. by providing assistance for piercing through tissue).

The knot-tying task was demonstrated with the system depicted in fig.1. Three grippers are employed to perform the knot. Two of them are controlled by the input instruments, one is autonomously controlled and keeps the loose end of the surgical thread under tension (in order to preserve a well-defined position). The trajectories of all grippers are stored in a data base for later evaluation. Afterwards, the matching algorithm of the scaffolding framework is employed to derive primitives from the demonstrated trajectory. Fig. 5 shows the state of detection after all primitives from the first track (right hand) in fig. 4 have been detected. Once the relevant information of a valid demonstration is stored in the knowledge base, the corresponding task can be instantiated in a new environment. The application of the surgical knot to a new position is depicted in fig. 6. This trajectory was actually performed by the surgical system presented above and finally a surgical knot was produced (see video attachment: till $0min$:$51sec$ manual control to prepare initial posture, afterwards the trajectory of fig. 6 is performed). However, skill application did not produce a knot in all cases (success rate was 67% for twelve trials). Failures have been due to entanglements of the thread material and to the mechanical play of the instruments.

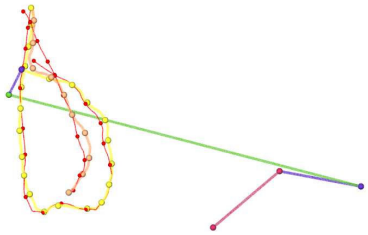


Fig. 5. **Detection of primitives**

## IV. Conclusion

We have based this approach of human-machine skill transfer on current psychological research on human communication and teaching. A methodology, which is particularly interesting for human-robot skill transfer, is situated learning, or more precisely, the scaffolding method. The basic idea of scaffolding is to utilize the superior knowledge of the teacher about the task to provide a framework of hints in order to assist the trainee. We have realized situated learning and scaffolding by means of a skill transfer architecture, which is based on abstract descriptions of tasks. Those are provided by the user and become an intrinsic part of the knowledge base of the system. The actual skill transfer is carried out via learning by demonstration, which by itself is an instantiation of situated learning. User demonstrations are decomposed into meaningful primitives with a finite automaton, which is constructed from the definitions in the scaffolding framework. The framework was successfully applied to the real-world application of robotic knot-tying. However, as mentioned above, the success rate of the system is still in need of improvement. One crucial point regarding this issue might be the control of the instrument servos. We are currently working on an improved version, which is based on brushless DC motors instead of pulse width modulated servos. Anyway, the current results mark already an improvement in comparison to the replay of raw demonstration data. Due to the limited absolute accuracy of our system (and probably every robotic system), this approach has always failed as soon as the difference between the positions of recording and replay exceeds a reasonable distance.

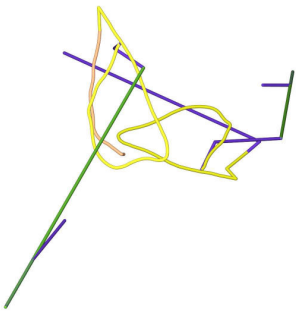


Fig. 6. **Task application in new environment**

## References


[1] J. Bigge, S. Best and K. Heller. Teaching Individuals with Physical, Health, or Multiple Disabilities (4th edition). Prentice Hall, Boston, USA, 2000.

[2] S. Jowsey. Can I Play Too?: Physical Education for Physically Disabled Children in Mainstream Schools. David Fulton Publishers, Oxford, UK, 1992.

[3] L. Vygotsky. Mind in Society: Development of Higher Psychological Processes (14th edition). Harvard University Press, Cambridge, MA, 1978.

[4] D. Wood and J. Bruner and G. Ross. The Role of Tutoring in Problem-Solving. *Journal of Child Psychology and Psychiatry*; Blackwell Publishing, vol. 17, pp. 89-100, 1976.

[5] C. Breazeal. Learning by Scaffolding. *Ph.D. thesis*, Massachusetts Institute of Technology, Cambridge, Maine, 1998.

[6] J. Saunders, C. Nehaniv, K. Dautenhahn and A. Alissandrakis. Self-Imitation and Environmental Scaffolding for Robot Teaching. *International Journal of Advanced Robotic Systems*; Ars International, vol. 4, no. 1, pp. 109-124, 2007.

[7] H. Mayer, I. Nagy, A. Knoll, E.U. Braun, R. Lange and R. Bauernschmitt. Adaptive Control for Human-Robot Skilltransfer: Trajectory Planning Based on Fluid Dynamics. *In Proceedings of the IEEE International Conference on Robotics and Automation*; Rome, Italy; 2007, pp. 1800-1807.

[8] G. Guthart and J. Salisbury. The Intuitive$^{TM}$Telesurgery System: Overview and Application, *In Proceedings of the IEEE International Conference on Robotics and Automation*; San Francisco, USA; 2000, pp. 618-621.

[9] K. Shoemake. Animating Rotation with Quaternion Curves. *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*; San Francisco, USA; 1985, pp. 245-254.

[10] S. Schaal, J. Peters, J. Nakanishi and A. Ijspeert. Learning movement primitives, *Springer tracts in Advanced Robotics: International Symposium on Robotics Research (ISRR)*; Siena, Italy; 2004 rec-nr. 1805.

[11] M. Matarić. Sensory-Motor Primitives as a Basis for Learning by Imitation: Linking Perception to Action and Biology to Robotics. Imitation in Animals and Artifacts, K. Dautenhahn and C. Nehaniv, editors; MIT Press, Cambridge, MA, 2002, pp. 392-422.

[12] K. Kleinmann, D. Bettenhausen and M. Seitz. A Modular Approach for Solving the Peg-in-Hole Problem with a Multifingered Gripper. *In Proceedings of the IEEE International Conference on Robotics and Automation*, Nagoya, Japan; 1995, pp. 758-763.

[13] M. Kaiser and R. Dillmann. Building elementary skills from human demonstration, *In: Proceedings of the IEEE International Conference on Robotics and Automation*; Minneapolis, Minnesota, USA; 1996, pp. 2700-2705.

[14] C. Hwang and K. Sasaki. Control Program of Two-Fingered Dexterous Manipulation with Primitive Motions. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*; Las Vegas, USA; 2003, pp. 2902-2907.

[15] K. Ogawara, J. Takamatsu, H. Kimura and K. Ikeuchi. Estimation of Essential Interaction from Multiple Demonstrations; *In Proceedings of the IEEE International Conference on Robotics and Automation*; Taipei, Taiwan; 2003, pp. 3893-3898.

[16] J.Takamatsu, T. Morita, K. Ogawara, H. Kimura and K. Ikeuchi. Representation for Knot-Tying Tasks. *IEEE Transaction on Robotics*; IEEE Robotics and Automation Society, vol. 22, no. 1, pp. 65-78, 2006.

[17] P. Hynes, G. Dodds and A. Wilkinson. Uncalibrated visual-servoing of a dual-arm robot for MIS suturing. *In: Proceedings of the IEEE International Conference on Biomedical Robotics and Biomechatronics*; Pisa, Italy; 2006, pp. 204-209.

[18] M. Kitagawa, A. Okamura, B. Bethea, V. Gott and W. Baumgartner. Analysis of suture manipulation forces for teleoperation with force feedback. *In Proceedings of the Fifth International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), T. Dohi and R. Kikinis (Eds.), Lecture Notes in Computer Science*; Springer-Verlag, vol. 2488, pp. 155-162, 2002.

[19] A. Kapoor, M. Li and R. Taylor. Spatial Motion Constraints for Robot Assisted Suturing Using Virtual Fixtures. *In Proceedings of the 8th International Conference on Medical Image Computing and Computer Assisted Intervention*; Palm Springs, USA; 2005, pp. 89-96.