In *Proceedings of the International Conference on Intelligent Robotics and Systems (IROS 2006)*

A System for Robotic Heart Surgery that Learns to Tie Knots Using Recurrent Neural Networks

Hermann Mayer[1], Faustino Gomez[2], Daan Wierstra[2], Istvan Nagy[1], Alois Knoll[1], and Jürgen Schmidhuber[1,2]
[1]Department of Embedded Systems and Robotics, Technical University Munich, D-85748 Garching, Germany
{mayerh|nagy|knoll|schmidhu}@in.tum.de
[2]Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), CH-6928 Manno-Lugano, Switzerland
{tino|daan|juergen}@idsia.ch

Abstract— **Tying suture knots is a time-consuming task performed frequently during Minimally Invasive Surgery (MIS). Automating this task could greatly reduce total surgery time for patients. Current solutions to this problem replay manually programmed trajectories, but a more general and robust approach is to use supervised machine learning to smooth surgeon-given training trajectories and generalize from them. Since knot-tying generally requires a controller with internal memory to distinguish between identical inputs that require different actions at different points along a trajectory, it would be impossible to teach the system using traditional feedforward neural nets or support vector machines. Instead we exploit more powerful, *recurrent* neural networks (RNNs) with adaptive internal states. Results obtained using LSTM RNNs trained by the recent Evolino algorithm show that this approach can significantly increase the efficiency of suture knot tying in MIS over preprogrammed control.**

I. Introduction

Minimally Invasive Surgery (MIS) has become commonplace for an ever-growing number of procedures. Because MIS is performed through small incisions or ports in the patient's body, tissue trauma, recovery time, and pain are reduced considerably compared to conventional, "open" surgery. While patients have profited enormously, surgeons have had to cope with reduced dexterity and perception: the instruments are long and have fewer degrees of freedom, force and tactile feedback are lost, and visual feedback is flattened to a 2D image. These factors make delicate maneuvers such as knot-tying very time-consuming. A laparoscopically tied suture knot can take up to three minutes to complete, compared to one second for a manually tied knot.

Robot-assisted MIS seeks to restore the feel of normal surgery by providing the surgeon with a more intuitive and ergonomic interface. The surgeon tele-operates a *slave* robot that manipulates the surgical instruments from a *master* console that provides full six degrees of freedom manipulation, enhanced 3D imaging, and often force feedback. Robotic surgical systems such as DaVinci [1] and ZEUS [2] are in wide use today, performing a variety of abdominal, pelvic, and thoracic procedures. However, despite significant advances in robot-assisted surgery, delicate tasks like knot-tying are still cumbersome and time-consuming, in some cases taking longer than with conventional MIS [3]. Given that knot-tying occurs frequently during surgery, automating this subtask would greatly reduce surgeon fatigue and total surgery time.

Building a good knot-tying controller is difficult because the 3D trajectories of multiple instruments must be precisely controlled. There has been very little work in autonomous robotic knot-tying: Kang et al. [4] devised a specialized stitching device, while Mayer et al. [5] were the first to tie a suture knot autonomously using general purpose laparoscopic instruments. In both approaches, the controller uses a hard-wired policy, meaning that it always repeats the same prescribed motion without the possibility of generalizing to unfamiliar instrument locations. One possible way to provide more robust control is to *learn* the control policy from examples of correct behavior, provided by the user.

The focus of this paper is on automating suture knot winding by training a recurrent neural network (RNN; [6]–[8]) on human generated examples. Unlike standard non-recurrent machine learning techniques such as support vector machines and feedforward neural networks, RNNs have an internal state or *short-term memory* which allows them to perform tasks such as knot-tying where the previous states (i.e. instrument positions) need to be remembered for long periods of time in order to select future actions appropriately.

To date, the only RNN capable of using memory over sequences the length of those found in knot-tying trajectories (over 1000 datapoints), is Long Short-Term Memory (LSTM [9]). Therefore, our experiments use this powerful architecture to learn to control the movement of a real surgical manipulator to successfully tie a knot. Best results were obtained using the recent hybrid supervised/evolutionary learning framework, Evolino [10], [11].

The next section describes the EndoPAR robotic system used in the experiments. In section III, we give a detailed account of the steps involved in laparoscopic knot tying. Section IV describes the Evolino framework, and in section V the method is tested experimentally in the task of autonomous suture knot winding.

II. The EndoPAR System

The Endoscopic Partial-Autonomous Robot (EndoPAR) system is an experimental robotic surgical platform developed by the Robotics and Embedded Systems research group at the Technical University of Munich (figure 1). EndoPAR consists of four Mitsubishi RV-6SL robotic arms that are mounted upside-down on an aluminum gantry, providing a 20cm$\times$25cm$\times$40cm workspace that is large enough for surgical procedures. Although there are four robots, it is easy to access the workspace due to the ceiling mounted setup. Three of the arms are equipped with force-feedback instruments; the fourth holds a 3D endoscopic stereo camera.

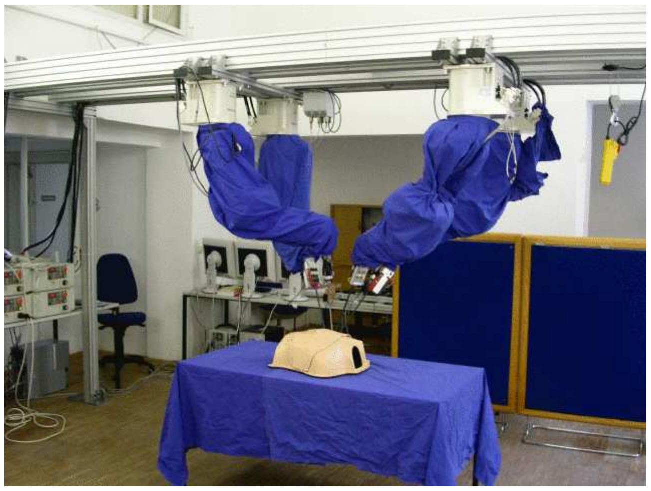

Fig. 1. **The EndoPAR system**. The four ceiling mounted robots are shown with an artificial chest on the operating table to test tele-operated and autonomous surgical procedures. Three of the robots hold laparoscopic gripper instruments, while the fourth manipulates an endoscopic stereo camera that provides the surgeon with images from inside the operating cavity. The size of the operating area (including gantry) is approximately 2.5m x 5.5m x 1.5m, and the height of the operating table is approximately 1 meter.

The position and orientation of the manipulators are controlled by two PHANToM^{TM} Premium 1.5 devices from Sensable Inc. The user steers each instrument by moving a stylus pen that simulates the hand posture and feel of conventional surgical implements. The key feature of the PHANToM devices is their ability to provide force feedback to the user. EndoPAR uses a version of the PHANToM device that can display forces in all translational directions (no torque is fed back).

Figure 2 shows the sensor configuration used to implement realistic force feedback in the EndoPAR system. Each instrument has four strain gauge sensors attached at the distal end of the shaft, i.e. near the gripper. The sensors are arranged in two full bridges, one for each principal axis. The signals from the sensors are amplified and transmitted via CAN-bus to a PC system where they are processed and sent to small servo motors that move the stylus to convey the sensation of force to the user. Since direct sensor readings are somewhat noisy, a smoothing filter is applied in order to stabilize the results.

Force feedback makes performing MIS more comfortable, efficient, safe, and precise. For knot-tying, this capability is essential due to the fine control required to execute the procedure without breaking or loosing the thread [12]. As a result, the EndoPAR system provides an excellent platform with which to generate good training samples for the supervised machine learning approach explored in this paper.

III. MIS Knot-Tying

Tying a suture knot laparoscopically involves coordinating the movements of three grippers through six steps. When the procedure begins, the grippers should be in the configuration depicted in figure 3A with the needle already having pierced the tissue (for safety, the piercing is performed manually by the surgeon). The next step (figure 3B) is to grasp the needle with gripper 1, and manually feed the thread to gripper 3,

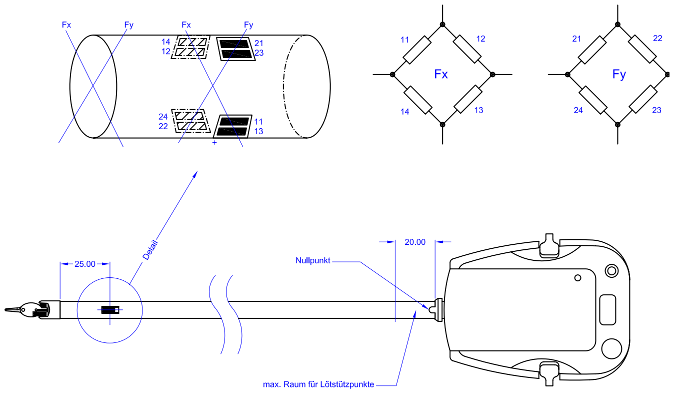

Fig. 2. **Force feedback**. Forces are measured in the x and y directions (perpendicular to shaft). The upper part of the figure shows how the strain gauge sensors are arranged along the circumference of the shaft. Each diametrically opposed pair constitutes a full bridge of four resistors dedicated to one principal axis. Sensor signals are sent back to servo motors at the input stylus so that the surgeon can sense forces occurring at the gripper.

the assistant gripper, making sure the thread is taut. Gripper 1 then pulls the thread through the puncture (figure 3C), while gripper 3 approaches it at the same speed so that the thread remains under tension. Meanwhile, gripper 2 is opened and moved to the position where the winding should take place.

Once gripper 2 is in position, gripper 1 makes a loop around it to produce a noose (figure 3D). For this step it is very important that the thread be under the right amount of tension; otherwise, the noose around gripper 2 will loosen and get lost. To maintain the desired tension, gripper 3 is moved towards the puncture to compensate for the material needed for winding. Special care must be taken to ensure that neither gripper 1 nor the needle interfere with gripper 2 or the strained thread during winding.

After completing the loop, gripper 2 can be moved to get the other end of the thread (figure 3E). Once again, it is critical that the thread stay under tension by having grippers 1 and 3 follow the movement of gripper 2 at an appropriate speed. In figure 3F, gripper 2 has grasped the end of the thread. Therefore, gripper 1 must loosen the loop so that gripper 2 can pull the thread end through the loop. Gripper 3 can now loosen its grasp, since thread tension is no longer needed. Finally, grippers 1 and 2 can pull outward (away from the puncture) in order to complete the knot.

The knot-tying procedure just described has been automated successfully by carefully programming the movement of each gripper directly [5]. Programming gripper trajectories correctly is difficult and time-consuming, and, more importantly, produces behavior that is tied to specific geometric coordinates. The next section describes a method that can potentially provide a more generic solution by learning directly from human experts.

IV. Evolino

Recurrent Neural Networks (RNNs) are a powerful class of models that can, in principle, approximate any dynamical system [13]. This means that RNNs can be used to implement arbitrary sequence-to-sequence mappings that require memory.

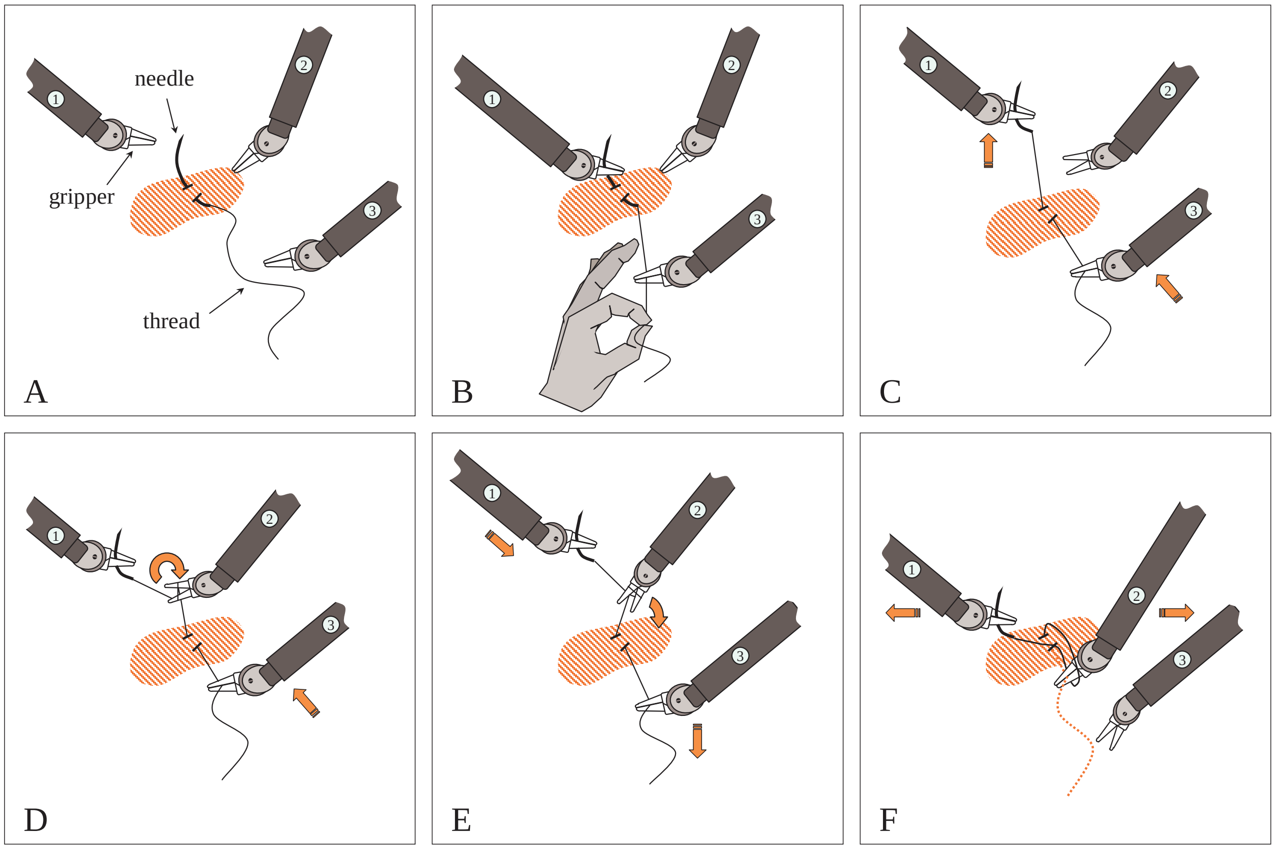

Fig. 3. **Minimally invasive knot-tying**. (A) The knot-typing procedure starts with the needle and three grippers in this configuration. (B) Gripper 1 takes the needle, and the thread is fed manually to gripper 3. (C) The thread is pulled through the puncture, and (D) wound around gripper 2. (E) Gripper 2 grabs the thread between the puncture and gripper 3. (F) The knot is finished by pulling the end of the thread through the loop.

However, training RNNs with standard gradient descent techniques is only practical when a short time window (less than 10 time steps) suffices to predict the correct system output. For longer time dependencies, the gradient vanishes as the error signal is propagated back through time so that network weights are never adjusted correctly to account for events far in the past [14].

Long Short-Term Memory (LSTM; [9], [15], [16]) overcomes this problem by using specialized, linear *memory cells* that can maintain their activation indefinitely. The cells have input and output gates that learn to open and close at appropriate times, either to let in new information from outside and change the state of the cell, or to let activation out to affect other cells or the network's output. This cell structure enables LSTM to learn long-term dependencies across almost arbitrarily long time spans. However, in cases where gradient is of little use due to numerous local minima, LSTM becomes less competitive (as in the case of learning gripper trajectories).

An alternative approach to training LSTM networks is the recently proposed Evolution of systems with Linear Outputs (Evolino; [10], [11]). Evolino is a framework for supervised sequence learning that combines neuroevolution [17] (the evolution of artificial neural networks) for learning the recurrent weights, with linear regression for computing the output weights (see figure 4).

During evolution, Evolino networks are evaluated in two phases. In the first phase, the network is fed the training sequences (e.g. examples of human-performed knot-tying), and the activations of the memory cells are saved at each time step. At this point, the network does *not* have connections to its outputs. Once the entire training set has been seen, the second phase begins by computing the output weights analytically using the pseudoinverse. The training set is then fed to the network again, but now the network propagates the input all the way through the new connections to produce an output signal. The error between the output and the correct (target) values is used as a fitness measure to be minimized by evolutionary search.

The particular instantiation of Evolino in this paper uses the Enforced SubPopulations algorithm (ESP; [18], [19]) to evolve LSTM networks. Enforced SubPopulations differs from standard neuroevolution methods in that instead of evolving complete networks, it *coevolves* separate subpopulations of network components or *neurons* (figure 4).

ESP searches the space of networks indirectly by sampling the possible networks that can be constructed from the subpopulations of neurons. Network evaluations provide a fitness statistic that is used to produce better neurons that can eventually be combined to form a successful network. This cooperative coevolutionary approach is an extension to Symbiotic, Adaptive Neuroevolution (SANE; [20]) which also evolves neurons, but in a single population. By using separate

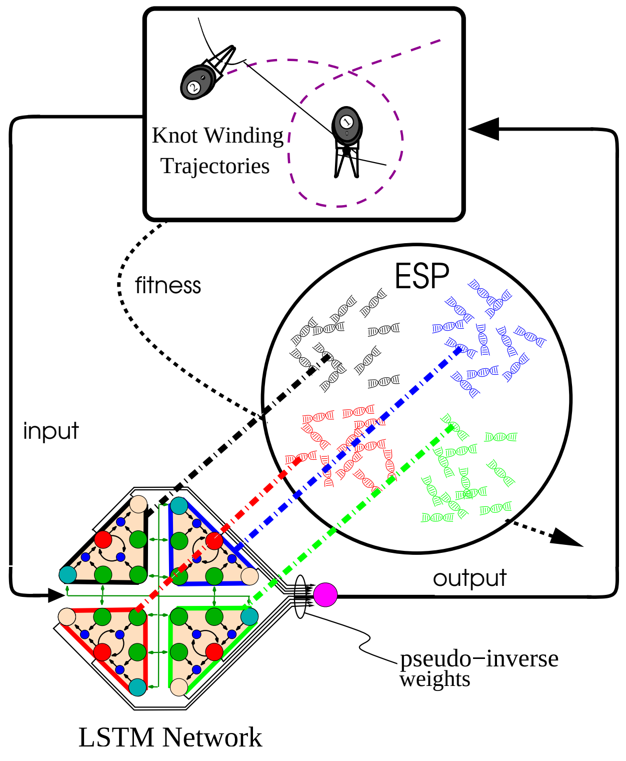

Fig. 4. **Evolino.** The figure shows the three components of the Evolino implementation used in this paper: the Enforced SubPopulations (ESP) neuroevolution method, the Long Short-Term Memory (LSTM) network architecture (shown with four memory cells), and the pseudoinverse method to compute the output weights. When a network is evaluated, it is first presented the training set to produce a sequence on network activation vectors that are used to compute the output weights. Then the training set is presented again, but now the activation also passes through the new connections to produce outputs. The error between the outputs and the targets is used by ESP as a fitness measure to be minimized.

subpopulations, ESP accelerates the specialization of neurons into different sub-functions needed to form good networks because members of different evolving sub-function types are prevented from mating. Subpopulations also make the neuron fitness evaluations less noisy because each evolving neuron type is guaranteed to be represented in every network that is formed. Consequently, ESP is able to evolve recurrent networks more efficiently than SANE.

Evolino does not evolve complete networks but rather evolves networks that produce a set of activation vectors that form a non-orthogonal basis from which an output mapping can easily be computed. The intuition is that it is often easier to find a sufficiently good basis than to find a network that models the target system directly. Evolino has been shown to outperform gradient-based methods on continuous trajectory generation tasks [10]. Unlike gradient-based methods, it has the ability to escape local minima due to its evolutionary component. Moreover, it is capable of generating precise outputs by using the pseudoinverse, which computes an optimal linear mapping. Previous work with Evolino has concentrated on comparisons with other methods in rather abstract benchmark problems, such as the Mackey-Glass time-series. This paper

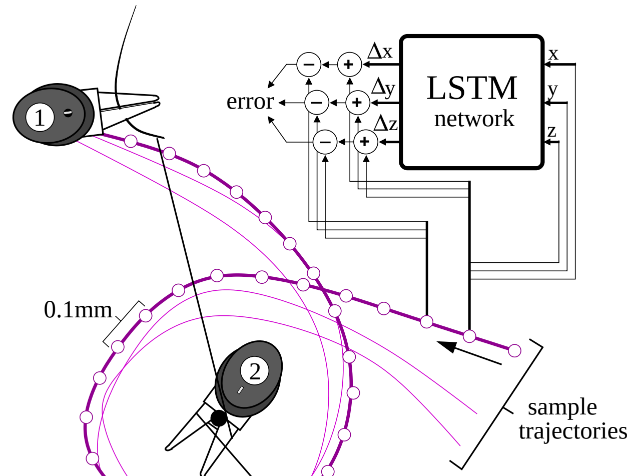

Fig. 5. **Training the knot winding networks**. LSTM networks are trained on a set of recordings that sample the position of gripper 1 at 0.1mm increments during a human-controlled suture knot. The figure shows three such training sequences; the one with the thicker path shows the sample points that the network uses as input and targets. For each training sequence, the network receives the (x, y, z)-position of the gripper 1, and outputs a prediction of the distance to the next position of the gripper (i.e. the next sample in the sequence). The prediction is added to the input and compared to the correct (target) next position to produce an error signal that is used either for gradient descent learning, or as a fitness measure for Evolino, after all the training sequences have been processed.

presents the first application of Evolino to a real-world task.

V. Experiments in Robotic Knot Winding

Our initial experiments focus on the most critical part of suture knot-tying: winding the suture loop (steps C through F in figure 3). While the loop is being wound by gripper 1, gripper 2 stays fixed. Therefore, networks were trained to control the movement of gripper 1.

A. Experimental Setup

LSTM networks were trained using a database of 25 loop trajectories generated by recording the movement of gripper 1 while a knot was being tied successfully using the PHANToM units. Each trajectory consisted of approximately 1300 gripper (x, y, z)-positions measured at every 0.1mm displacement, $\{(x^j_1, y^j_1, z^j_1), \ldots, (x^j_{l_j}, y^j_{l_j}, z^j_{l_j})\}, j = 1..25$, where l_j is the length of sequence j. At each step in a training sequence, the network receives the coordinates of gripper 1 through three input units (plus a bias unit), and computes the desired displacement $(\Delta x, \Delta y, \Delta z)$ from the previous position through three output units.

Both gradient descent and Evolino were used in 20 experiments each to train LSTM networks with 10 memory cells. The Evolino-based networks were evolved for 60 generations with a population size of 40, yielding a total of 3580 evaluations (i.e. passes through the training set) for each experiment.

Figure 5 illustrates the procedure for training the networks. For the gradient descent approach, the LSTM networks were trained using Backpropagation Through Time [6] where the network is unfolded once for each element in the training sequence to form an l_j-layer network (for sequence j) with all layers sharing the same weights. Once the network has seen that last element in the sequence, the errors from each time-step are propagated back through each layer as in standard backpropagation, and then the weights are adjusted.

For Evolino-trained LSTM, each network is evaluated in two phases (see section IV). In the first phase the activations of the network units are recorded, but no outputs are produced as, at this point, the network does not have output connections. After the entire training set has been seen, the output connections are computed using the pseudoinverse. In the second phase, the network produces control actions that are used to calculate the fitness of the network.

The error (fitness) measure used for both methods was the sum-squared difference between the network output plus the previous gripper position and the correct (target) position for each time-step, across the 25 trajectories:

$$\sum_{j=1}^{25}\sum_{t=1}^{l_j-1}(x_t^j+\Delta x_t-x_{t+1}^j)^2+(y_t^j+\Delta y_t-y_{t+1}^j)^2+(z_t^j+\Delta z_t-x_{t+1}^j)^2$$

where $\Delta x_t, \Delta y_t$, and Δz_t are the network outputs for each principal axis at time t which are added to the current position (x_t, y_t, z_t) to obtain the next position. Note that because the networks are recurrent, the output can in general depend on all of the previous inputs.

For the first 50 time-steps of each training sequence, the network receives the corresponding sequence entry. After that, the network feeds back its current output plus its previous input as its new input for the next time-step. That is, after a *washout time* of 50 time-steps, the network makes predictions based on previous predictions, having no access to the training set to steer it back on course. This procedure allows the error to accumulate all along the trajectory so that minimizing it forces the network to produce loop trajectories autonomously (i.e. in the absence of a "teacher" input).

Once a network has learned the training set, it is tested in a 3D simulation environment to verify that the trajectories do not behave erratically or cause collisions. If the network passes this validation, it is transferred to the real robot where the procedure is executed inside the artificial rib-cage and heart mockup shown in figure 1. To tie the entire knot, a preprogrammed controller is used to start the knot, and then the network takes over for the loop, steps C through E. During the loop, the robot fetches a new displacement vector from the network every 7ms, and adds it to the current position of gripper 1. Gripper 2 remains stationary throughout this phase, and gripper 3 is moved away from the knot at a predefined rate to maintain tension on the thread. When the loop is complete, the control switches back to the program to close the knot. As in training, the winding network receives an initial "approaching sequence" of 50 points that control the robot to start the wind, and then completes the loop itself while feeding back its own outputs.

B. Experimental Results

Figure 6 shows the learning curve for the Evolino-trained LSTM networks. Each datapoint is the average error on the training set of the best network, measured in millimeters. By generation 40, the error has reached a level that the networks can effectively produce usable loop trajectories. The gradient-trained LSTM networks were not able to learn the trajectories, so the error for this method is not reported. This poor performance could be due to the presence of many local minima in the error surface which can trap gradient-based methods.

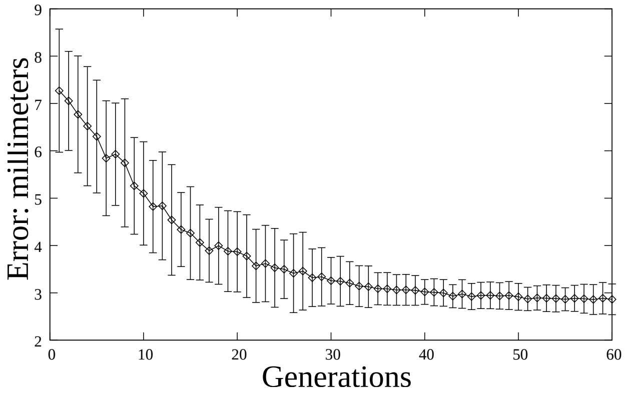

Fig. 6. **Evolino learning curve**. The plot shows the average error on the training set measured in millimeters for the best network in each generation, averaged over 50 runs. The vertical bars indicate one standard deviation from the average.

Unlike gradient-based approaches, Evolino is an evolutionary method, and therefore is less susceptible to local minima. All of the 20 Evolino runs produced networks that could generate smooth loop trajectories. When tested on the real robot, the networks reliably completed the loop procedure, and did so in an average of 3.4 seconds, a speed-up of almost four times over the preprogrammed loop. This speed-up in knot winding results in a total time of 25.8 sec for the entire knot, compared to 33.7 sec for the preprogrammed controller.

Figure 7 shows the behavior of several Evolino-trained LSTM networks from the same run at different stages of evolution. As evolution progresses, the controllers track the training trajectories more closely while smoothing them. The network in the right-hand side of the figure was produced after approximately 4.5 hours of computation time.

These first results show that RNNs can be used to learn from training sequences of over one thousand time steps, and possibly provide useful assistance to expedite MIS procedures.

VI. Discussion and Future Work

An important advantage of learning directly from expert behavior is that it requires less knowledge about the system being controlled. Supervised machine learning can be used to capture and generalize expertise without requiring the often tedious and costly process of traditional controller design. The Evolino-trained LSTM networks in our experiments were able to learn from surgeons and outperform them on the real robot.

Our current approach only deals with the winding portion of the knot-tying task. Therefore, its contribution is limited by the efficiency of the other subtasks required to complete the full knot. In the future, we plan to apply the same basic approach used in this paper to other knot-tying subtasks (e.g. the thread tensioning performed by the assistant gripper, and knot

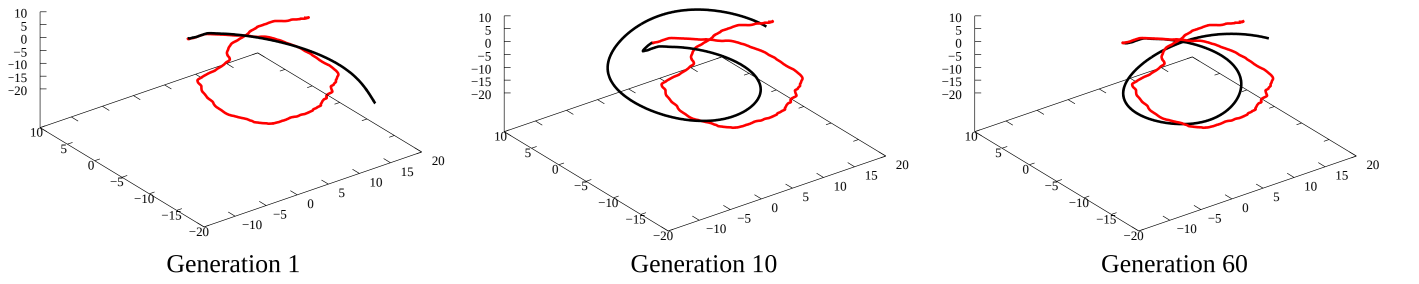

Fig. 7. **Evolution of loop generating behavior.** Each of the three 3D plots shows the behavior of the best network at a different generation during the same evolutionary run. All axes are in millimeters. The dark curve is the trajectory generated by the network; the lighter curve is the target trajectory. Note that the network is being tested to reproduce same target trajectory in each plot. The best network in the first generation tracks the target trajectory closely for the first 15mm or so, but diverges quickly when the target turns abruptly. By the tenth generation, networks can form smooth loops, and by generation 60, the network tracks the target throughout the winding, forming a tight, clean loop.

tightening) that are currently implemented by programmed controllers. The separate sub-controllers can then be used in sequence to complete the whole procedure.

The performance of automated MIS need not be constrained by the proficiency of available experts. While human surgeons provide the best existing control, more optimal strategies may be possible by employing reinforcement learning techniques where target trajectories are not provided, but instead some higher-level measure of performance is maximized. Approaches such as neuroevolution could be used alone, or in conjunction with supervised learning to bootstrap the learning. Such an approach would first require building a simulation environment that accurately models thread physics.

VII. Conclusion

This paper has explored the application of supervised learning techniques to the important task of automated knot-tying in Minimally Invasive Surgery. Long Short-Term Memory neural networks were trained to produce knot winding trajectories for a robotic surgical manipulator, based on human-generated examples of correct behavior. Initial results using the Evolino framework to train the networks are promising: the networks were able to perform the task on the real robot without access to the teaching examples. These results constitute the first successful application of supervised learning to MIS knot-tying.

Acknowledgments

This research was partially funded by SNF grant 200020-107534 and the EU MindRaces project FP6 511931.

References

[1] G. Guthart and J. K. S. Jr., "The Intuitive^{TM} telesurgery system: Overview and application," in *International Conference on Robotics and Automation (ICRA)*, 2000, pp. 618–621.

[2] A. Garcia-Ruiz, N. Smedira, F. Loop, J. Hahn, C. Steiner, J. Miller, and M. Gagner, "Robotic surgical instruments for dexterity enhancement in thorascopic coronary artery bypass graft," *Journal of Laparoendoscopic and Advanced Surgical Techniques*, vol. 7, no. 5, pp. 277–283, 1997.

[3] A. Garcia-Ruiz, "Manual vs robotically assisted laparoscopic surgery in the performance of basic manipulation and suturing tasks," *Archives of Surgery*, vol. 133, no. 9, pp. 957–961, 1998.

[4] H. Kang, "Robotic assisted suturing in minimally invasive surgery," Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, New York, May 2002.

[5] H. Mayer, I. Nagy, A. Knoll, E. U. Schirmbeck, and R. Bauernschmitt, "The EndoPAR system for minimally invasive surgery," in *International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.

[6] P. Werbos, "Backpropagation through time: what does it do and how to do it," in *Proceedings of IEEE*, vol. 78, 1990, pp. 1550–1560.

[7] A. J. Robinson and F. Fallside, "The utility driven dynamic error propagation network," Cambridge University Engineering Department, Tech. Rep. CUED/F-INFENG/TR.1, 1987.

[8] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] J. Schmidhuber, D. Wierstra, and F. Gomez, "Evolino: Hybrid neuroevolution/optimal linear search for sequence learning," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 2005.

[11] D. Wierstra, F. Gomez, and J. Schmidhuber, "Modeling non-linear dynamical systems with evolino," in *Proceedings of the Genetic Evolutionary Computation Conference (GECCO-05)*. Berlin; New York: Springer-Verlag, 2005.

[12] I.Nagy, H. Mayer, A. Knoll, E. Schirmbeck, and R. Bauernschmitt, "EndoPAR: An open evaluation system for minimally invasive robotic surgery," in *IEEE Mechatronics and Robotics 2004 (MechRob)*, Aachen, Germany, September 2004.

[13] H. T. Siegelmann and E. D. Sontag, "Turing computability with neural nets," *Applied Mathematics Letters*, vol. 4, no. 6, pp. 77–80, 1991.

[14] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*, S. C. Kremer and J. F. Kolen, Eds. IEEE Press, 2001.

[15] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[16] F. A. Gers and J. Schmidhuber, "LSTM recurrent networks learn simple context free and context sensitive languages," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333–1340, 2001.

[17] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[18] F. Gomez and R. Miikkulainen, "Solving non-Markovian control tasks with neuroevolution," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. Denver, CO: Morgan Kaufmann, 1999. [Online]. Available: http://nn.cs.utexas.edu/keyword?gomez:ijcai99

[19] F. J. Gomez and R. Miikkulainen, "Active guidance for a finless rocket using neuroevolution," in *Proc. GECCO 2003, Chicago*, 2003, *Winner of Best Paper Award in Real World Applications*.

[20] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Machine Learning*, vol. 22, pp. 11–32, 1996.