# Skill Transfer and Learning by Demostration in a Realistic Scenario of Laparoscopic Surgery

Hermann Mayer, István Nagy, Alois Knoll

Technische Universität München, 85747 Garching, Germany,
nagy@in.tum.de,
mayerh@in.tum.de,
knoll@in.tum.de
WWW home page: http://www6.in.tum.de

**Abstract.** Commercially available systems for laparoscopic surgery usually come without operator side force feedback nor instrument-side force sensory. This often leads to increased trauma of tissue. Yet another reason why such systems are not widely accepted for every-day usage is a prolonged operation time, due to time consuming micro-manipulation tasks, e.g. knot tying. We present an approach which tackles both issues. On the one hand we provide a realistic surgical environment including haptic feedback, on the other hand we present basic techniques for partial autonomy. Autonomy will be implemented by combining two established methods of human-robot instruction: learning by demonstration and primitive instantiation. Once presented by the human instructor, subtasks are generalized and partitioned in reusable primitives, which can be recombined in order to solve a priori unknown tasks.

## 1 Hardware Setup

### 1.1 Overview

We describe in this section an experimental hardware setup, which we built in order to evaluate machine learning techniques in a realistic scenario of laparoscopic surgery. One main feature is high fidelity force feedback, which most commercially availale systems lack of. The system is yet powerful enough to serve as a prototypical implementation of a teleoperation system.

Figure 1 shows a snapshot of the system during operation. Two KUKA ([1]) industrial robots are carrying original instruments from Intuitive Surgical ([2]), as they are used in their $daVinci^{TM}$ system. A more detailed overview about this teleoperator can be found in [3]. We equipped the instruments with force sensory, which enables us to feed back real forces to the input devices, two Sensable PHANToMs with 3 DOF force feedback. Both PHANToMs are directly connected to the corresponding KUKA control cabinets, so that a real-time remote control is possible. As a future extension we plan to mount a stereo-view capable camera system on a third robot in order to achieve conditions similar to a real telepresence system.

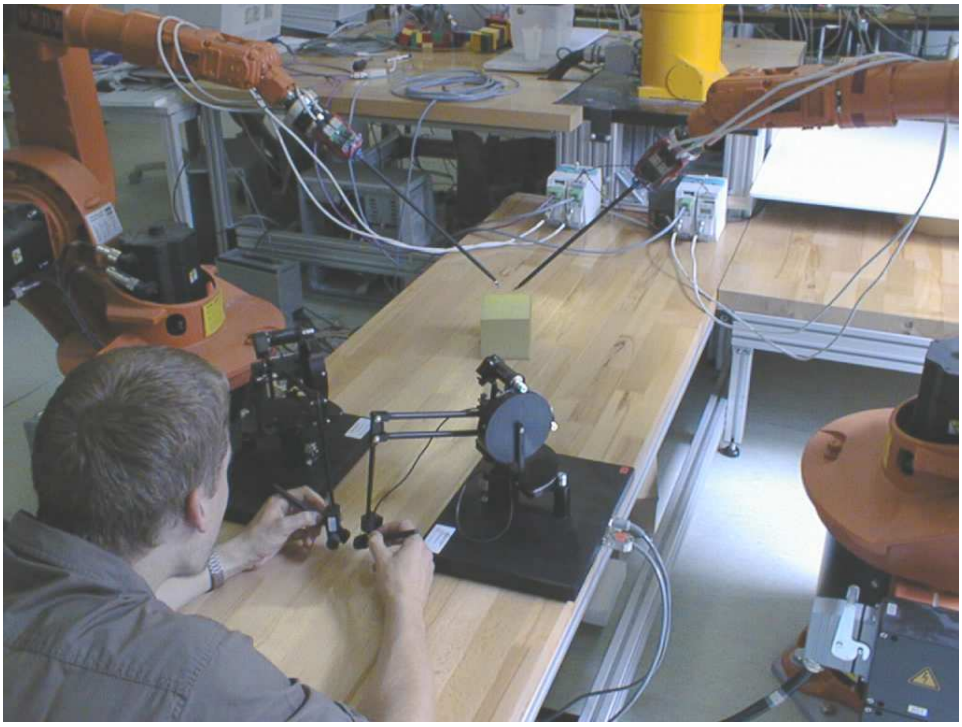

**Fig. 1.** Experimantal hardware setup

### 1.2 Force Sensory Loop

We now give a detailed description of the force sensory and feedback subsystem. As depicted in Figure 2 it comprizes the two PHANToMs, four amplifiers and strain gauges applied to surgical instruments. All devices are connected to a single Linux PC. While the PHANToMs are plugged into PCI interfaces, the force sensory parts are interconnected via DeviceNet (real-time capable industrial bus system). The strain gauges are directly applied to the shaft of the instruments, near to the micro-gripper. They are arranged perpendicular to each other forming a full measuring bridge per degree of freedom. At the moment no sensory is installed which enables measurement of longitudinal forces along the shaft, but will be available in the future. The construction is sensitive enough to provide high fidelity kinesthetics. Figure 3 shows a schematic view of the sensor configuration. Each sensor reading is fed into an PME MP 30 amplifier supplied by HBM ([4]). They are theoretically capable of 1 kHz sampling rate, but in our interconnected configuration only 750 Hz are reached due to high bus traffic. These versatile lab amplifiers can of course be replaced by cheaper off the shelf components of reduced size as they are used in standard industrial setups.

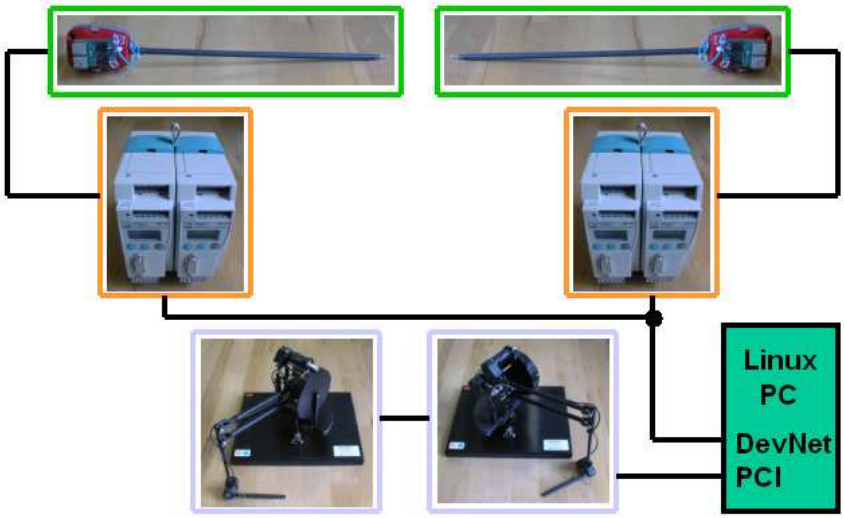


**Fig. 2.** Schematic view of force sensory loop

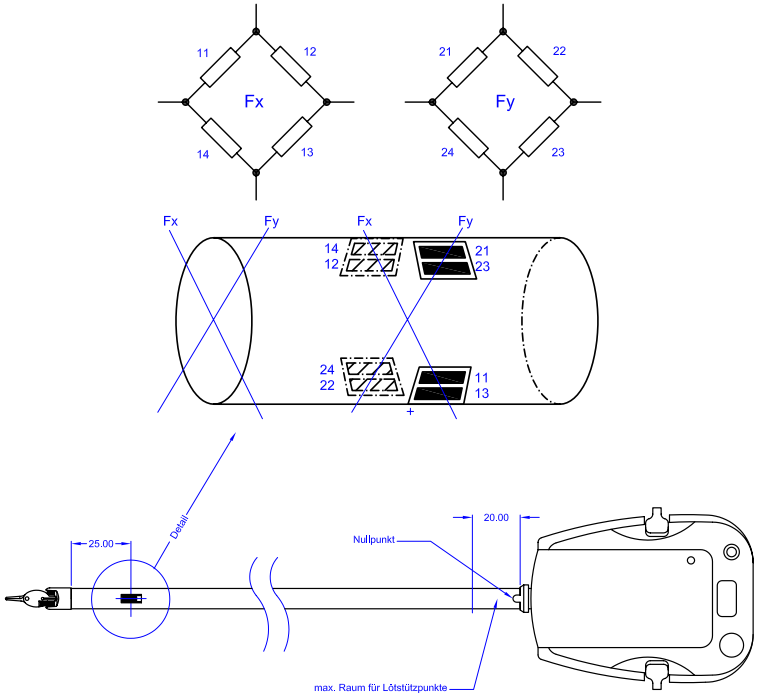


**Fig. 3.** Strain gauge application blueprint

To reflect the measured forces to the operator we rely on he SDK delivered with the PHANToM devices. The software allows not only the creation of simulated haptic environments, but also the representation of "external" forces. This is done by means of subclassing existing force field objects. Additionally it facilitates the integration of 3D graphics, which enables us to easily create a simulation environment.

## 2 Skill Transfer

In the previous section we described an experimental setup which enables the verification of basic learning by demonstration techniques in realistic scenarios of laparoscopic surgery. We picked out knot tying as one specific setting, which we think best characterizes the typical problem statements. Our experiences are, that this task is very time consuming and complex, which was also mentioned in [5]. It's understood that instrument knots (see fig. 4) are meant, whose execution significantly differs from conventional surgical knots.

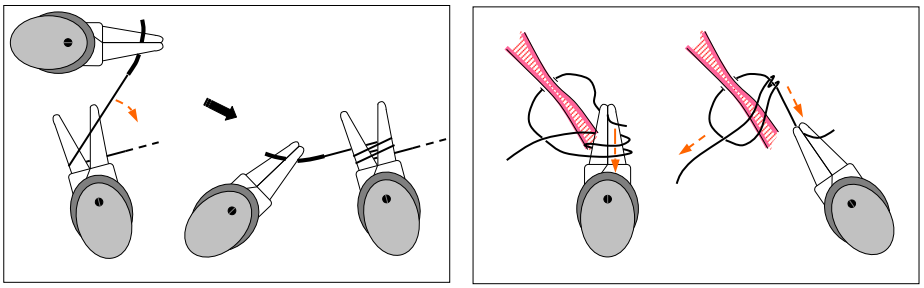

**Fig. 4.** Tying a knot with instruments

The integration of force sensory and force feedback allows us to incorporate additional haptical information in our learn strategies yielding to a multimodal skill transfer, which is indispensable in this context. A high level description of a typical operating sequence looks as follows:

1. A human operator executes different instrument knots, a few of each.
2. The system learns (generalizes) in an unsupervised manner.
3. The system should be able to complete already known manipulation sequences as the human operator initiates them.

Our implementing software architecture comprizes three hierarchical layers. The basic layer is data acquisition and preprocessing. Due to physiological tremor of the human operator and noisy sensor readings, raw data filtering is necessary. After data preprocessing, manipulation sequences are split into their primitives. We use two different approaches. On the one hand spline features like curvature and torsion, on the other hand sequence alignment technologies adapted from

bioinformatics. To learn higher level skills from primitives we are planning to implement methods known from optical character recognition and cluster analysis.

### 2.1 Data Acquisition and Preprocessing

The raw data describing a complex manipulation sequence consists of the instrument positions and orientations, the forces occured and additionally the state of the instrumental side microgripper. It is obvious that the data has to be preprocessed before it can be used by subsequent steps.

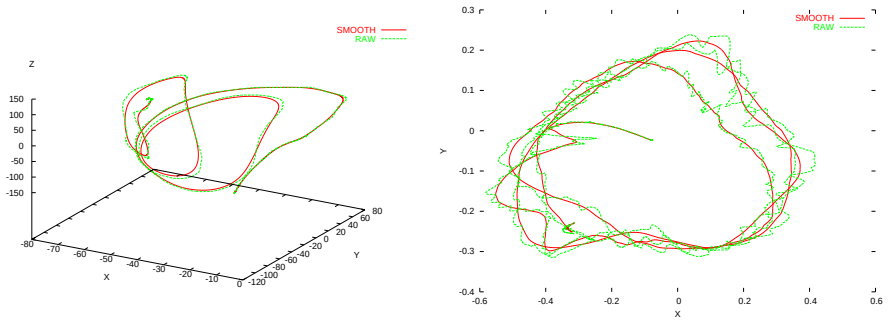


**Fig. 5.** Raw and smoothed data: (a) Positions; (b) Forces

There is an amount of noise due to the inevitable physiological tremor of the human operator, furthermore we have high hardware-side sample rates for acquisitioning positions, orientations and forces, as they are needed for a realistic force feedback and robot control, but obstructive for machine learning. Positions and orientations are sampled (at hardware-side) with 1000 Hz (GHOST SDK), whereas forces with 750 Hz, so there is in both cases room for smoothing and lowpass filtering without information loss. Figure 5 shows raw and smoothed positions and forces of a simple winding process, which is one important subtask in knot tying. We have recorded at software-side with a sampling rate of 500 Hz, and took real suture material used in surgical practice to be as close as possible to reality. Postprocessing the force values seems to be even more necessary, but smoothing the trajectories using a sliding window average provides a good basis for the next processing steps. The size of the window over which local areas were averaged was 256 for the positions and 128 for the forces. Note that the forces are plotted 2D since we only have forces in $X$ and $Y$ directions yet, $Z$ direction sensory is in work now.

### 2.2 Low-Level Symbolic Representation

Due to the big amount of data it is not advisable to store complex manipulation sequences in their raw form. Therefore low-level symbolic representations are needed, which are on one hand inexpensive to compute, and on the other hand allow an efficient storage. Such representations have two further aims: the possibility of primitive decomposition and of defining features needed for the later matching process. It has been found that it is advantageous to interpret the multimodal data (positions, orientation, forces) representing a complex manipulation sequence as curves in 3D space, as they always have three components: POS$(x, y, z)$, ORI$(a, b, c)$, FORCE$(fx, fy, fz)$. In order to achieve more adjustable matching capabilities, two different techniques were used: spline interpolation and 3D chaincoding. Both techniques are computationally inexpensive and require much less storage place than the original data.

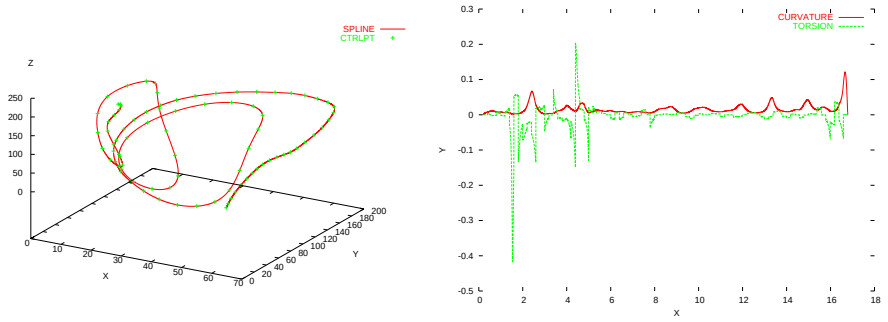


**Fig. 6.** (a) Spline interpolation; (b) Curvature and Torsion

**Spline interpolation** For the spline interpolation natural cubic splines were used. Figure 6a shows an interpolating spline of the smoothed position trajectory from figure 5a. The control points are the result of a simple resampling process, whereby also a translation to a positive quadrant took place for simpler chaincoding. Beside the fact, that natural cubic splines are the "smoothest" two times continuously differentiable interpolating functions, they also provide two important features: curvature and torsion. Curvature measures how sharply a curve is turning while torsion measures the extent of its twist in 3D space. Curvature and torsion (fig. 6b) completely define the shape of a 3D curve. Furthermore having a spline representation allows the extraction of equidistant data even if it wasn't recorded in that way, an important criterion for certain machine learning approaches.

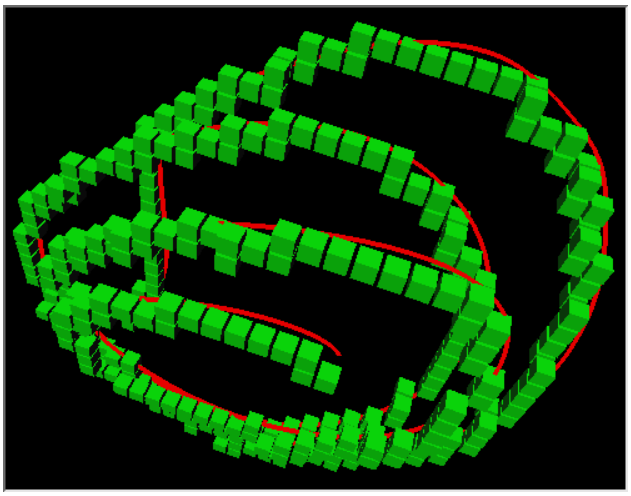

**Fig. 7.** 3D chaincoding

**3D Chaincoding** A method for representing digital curves using chain codes was first mentioned by Freeman in 1961 ([6], [7]). Since then many authors have been using techniques of chain coding due to the fact that various shape features may be computed directly from this representation. He also mentioned a method for representing 3D digital curves ([8]), his method however involved assigning a symbol for each of the 26 possible neighbourhood positions in 3D space. As opposed to Freeman's absolute directions, Bribiesca [9] considers relative direction changes, which allow to have a curve description that is invariant under translation and rotation.

In our case chaincoding in the initial coordinate system doesn't make much sense, a quantization on a cubic lattice is necessary. Figure 7 shows the chaincode representation of the spline in a cubic lattice of the size $[30 \times 30 \times 30]$, which is the result of a coarse resampling of the original bounding box of the curve. Therefore the spline was simply traversed, the properly rounded and scaled coordinates were used as indices to a 3D integer array to indicate whether a lattice cube contains a curve portion or not. There are a couple of shape properties and alternative representations for chaincodes, so that useful features describing a certain curve are available. The most important are the Fourier descriptors, which provide a means to characterize contours. The idea is to represent the contour as a function of one variable, expand the functions in terms of its Fourier series, and use the coefficients of the series as Fourier descriptors. Kuhl et al. investigated in [10] and [11] ways of obtaining the Fourier coefficients of Freeman encoded contours. The applicability of the 3D correspondants of features available for 2D chaincoded curves like the $\psi$-s curve, eccentricity, compactness, concavity, slope density function and shape numbers could also be evaluated.

## 3 Detecting Manipulation Primitives

In this chapter we want to describe two approaches to derive primitives from acquired data. It uses on one hand spline features like curvature and torsion, on the other hand an analogy to bioinformatics and conventional cluster analysis. These methods generate a symbolic representation of the input data which is independent from the actual demonstration of a skill. Each demonstrated skill is a combination of certain motion primitives. Our intention is to build up a library of primitives which can be recombined to skills in order to solve a wide range of problems. Primitives in this context are complex temporally ordered operating sequences that alter the status of objects in the application domain. A primitive can be described by means of paramaters determining the relative position of manipulated objects (e.g. the tool tip of a medical instrument), their orientation and occurring forces. Each parameter is a function defined over time and again on its part can be adjusted by certain parameters in order to apply the primitive to different situations. Additionally, the actual effectuation of a primitive is also dependent on external events that are detected by sensors (e.g. stop movement, if contact force detected). A possible primitive would be the wrapping of the thread around the pincer with constant force by avoiding contact with surrounding tissue.

In order to learn a skill, the system first has to observe a human operator demonstrating the skill. In our case the implementing units of both, operator and machine, are identical, because both control the same device (telemanipulator). In an extended scenario this identity can be abandoned, e.g. by observing a task manually performed by a human, while the machine executes learned skills with robotic hands. We pick up the idea of breaking down skills to primitives (see [12] or [13]) in order to reach a higher flexibility: primitives can be reused for different skills or even recombined to create an a priori unknown skill to solve a new problem. Therefore it is not only important to record the demonstrated skill, but also to classify the situation in which a certain skill is used. A situation is an episode in the multimodal input stream of the machine, comprising image processing, force sensing and reading positions. Our goal on the long run is to recognize certain situations in which a formerly learned skill can be applicated. This skill can be effectuated after approval of the operator.

As indicated above, the main design criteria of a primitive is reusability. I.e. it should be possible to reuse a primitive in a variety of tasks. On the other hand a skill should not be too trivial (e.g. movement on a straight line). Trivial primitives can be reused in a maximum number of skills (each movement can be approximated by a sequence of straight lines), but context dependency will be lost (the execution of a straight line movement cannot be mapped to a certain situation). Learning, or in other words mapping situations to skills, will be impossible. Therefore primitive decomposition is a tradeoff between fexibility and unambiguousness. In order to make allowance for this discrepancy, primitives are no static constructs, but can be adjusted as learning advances (e.g. alter parameters, split a primitive in two new ones etc.).

### 3.1 Spline Features

We have seen in the last section that a spline representation provides us with continuous curvature and torsion values at each point of an interpolated 3D curve. The idea is, that manipulation primitives begin (and close) at certain points in time, where "*anomalies*" occur in the spline representations of the multimodal data describing a complex manipulation sequence. It stands to reason, that very abrupt changes (therefore we call them "*anomalies*") both in positions/orientations and forces point to manipulation primitive boundaries.

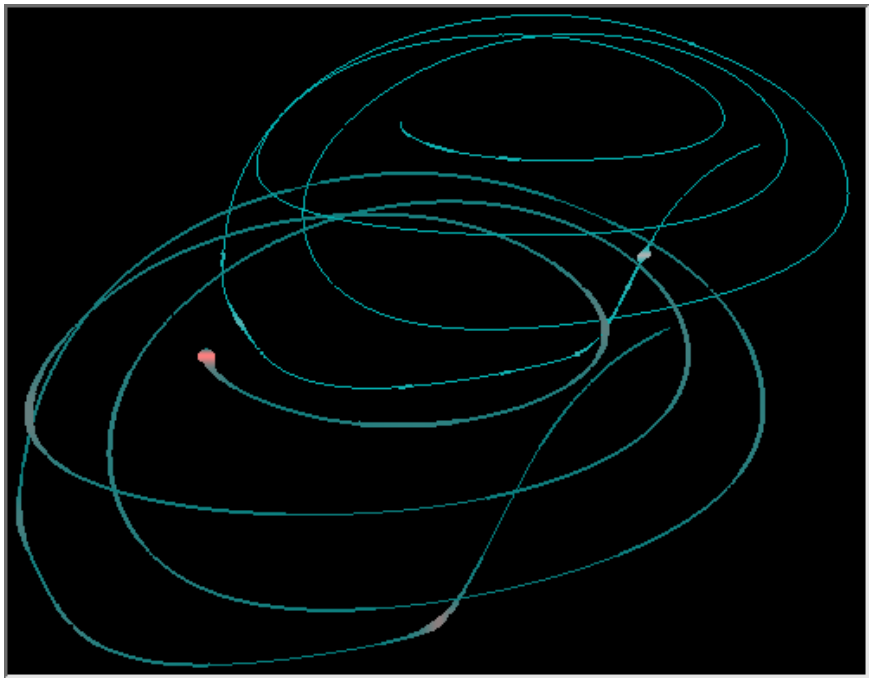

**Fig. 8.** Position trajectory spline augmented by curvature and torsion

Figure 8 shows the position trajectory spline augmented by curvature and torsion. The respective values are visualized by setting both color and thickness according to curvature and torsion on the curve progression, the lower-left spline is augmented by torsions, the upper-right one by curvatures.

### 3.2 Sequence Alignment with Trajectories

Once sampled equidistantly, data points of two different skills can now be scanned for similarities. Therefore we will try to match sections of different trajectories on

each other. This can be done by means of algorithms adopted from bioinformatics, because our problem is analog to the task of finding similar subsequences in genetic code. In bioinformatics we compare sequences by successively comparing atomic parts (e.g. DNA bases) of the structure. In addition we will only apply algorithms of bioinformatics which suppose that occurrences of atomic elements are independent from each other. This does not apply if we assume certain probabilities for mutations or the like.

In order to suit those algorithms to our problem, we first have to find a string-like representation of the data, consisting of atomic parts whose occurrences are independent from each other. For now we restrict our approach on regarding only the three dimensional position of a trajectory, while neglecting tool rotation and forces. These parameters will be included in a future version. In order to compare similar trajectories recorded at different positions and under various angles, we have to represent the data-points independently from any coordinate system, while still preserving the relative position of points to each other. One idea to reach this goal is to observe the concomitant trihedron of the trajectory at successive data-points (see fig. 9).

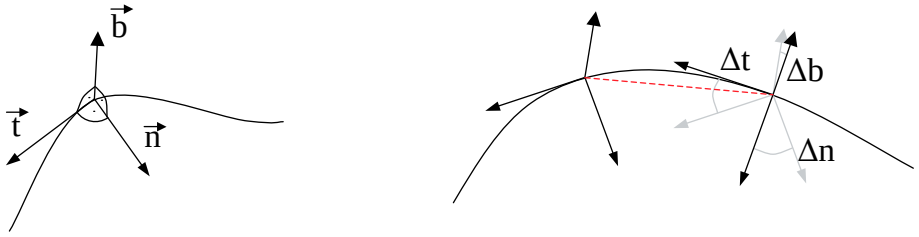


**Fig. 9.** Determining the difference between subsequent trihedrons

While each trihedron itself is expressed in terms of base coordinates, the differece between the trihedrons of successive points can be expressed independently from any coordinate system. We only want to regard the difference in rotation, while neglecting translations. This is acceptable, because we have already presampled all data points in a way that they have always the same distace to each direct neighbor. To simplify our first try with this method, we do not calculate the unique rotation matrix between two successive trihedrons, but only regard the angles between tangents $\Delta \boldsymbol{t}$ and normals $\Delta \boldsymbol{n}$ (see fig. 10 left side). Angles between binormals $\Delta \boldsymbol{b}$ can be neglected, because they will exhibit no additional information. Note that in general this approach is ambivalent, because we will use the inner product to calculate angles and therefore the direction of the curvature is neglected. Results have shown that this procedure is convenient for most real-world examples.

We now have a coordinate-independent representation of our trajectory and we can compare data points of different trajectories with each other (like it is done with DNA-sequences: see fig. 10). Next we select an appropriate algorithm

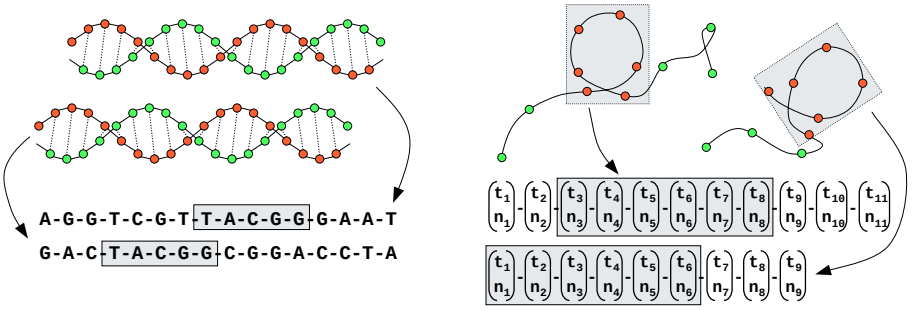


**Fig. 10.** Analogy between Bioinformatics and Robotics

to find similarities. Because matching parts could occur at any place within a trajectory, we will adapt an algorithm searching for local alignments in two strings. One algorithm that solves this problem efficiently and can be easily implemented, is the algorithm of Smith and Waterman. An extensive description of this algorithm can be found at [14] or [15]. In short, it defines an award for matching symbols, while punishing insertions, deletions and substitutions. A matrix is defined where each entry $m(i, j)$ denotes the costs for locally aligning $string_1$ (substring of the first sequence) ending with character $i$, with $string_2$ ending with character $j$. If we refer to the horizontally aligned sequence in figure 11, a local alignment can be constructed by the following operations:

1. assume a deletion in comparison with the other string (vertical movement in fig. 11)
2. assume an insertion (horizontal movement)
3. assume a substitution (diagonal movement)
4. a match was found (diagonal movement labeled with the same characters)

Each insertion or deletion is punished by a discount of -2, while each substitution is punished by -3. On the other hand, each match is rewarded by a gain of +3. In figure 11 the local alignment of the (sub)strings T-CG with TACG and TAAGG with TAAGG is shown (the dash denotes a deletion). In our case we cannot make any deletions or insertions, because the value at each data-point is defined by the difference of the corresponding trihedron and its successor. That means the correct course of the curve at a certain point can only be determined, if all relative changes of the curve at predecessing points are known. Therefore we have to reduce the scope of the algorithm to finding socalled diagonal runs (e.g. local alignment TAAGG in fig. 11). Another issue is the transfer of matches and substitutions to our application domain. Because we operate with floating point numbers we would rarely find exact matches and on the other hand it would not be adequate to speak of a mismatch or substitution if the difference between values is very small. In addition we have to deal with a value vector instead of only a single character. To cope with this, we have decided to use the following similarity function: $s = -25((n_1 - n_2)^2 + (t_1 - t_2)^2) + 1$, where $n_1$

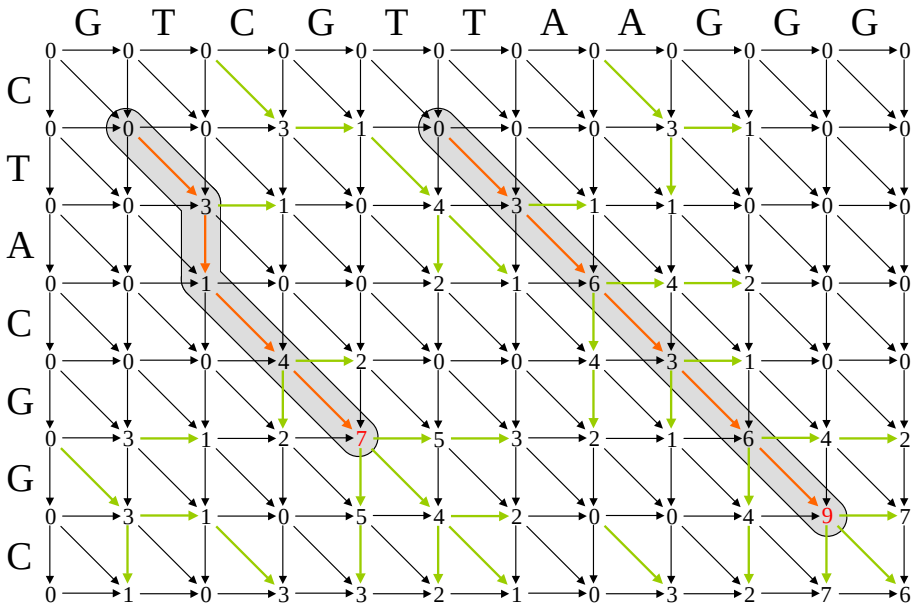


**Fig. 11.** Example for a local alignment search

and $t_1$ are the differences of angles between normals and tangents of successive trihedrons determined for the first trajectory. Accordingly, $n_2$ and $t_2$ are the values calculated for the second trajectory. The function $s$ was found by experience and it has turned out to provide good results in all test cases. The algorithm returns one or more subsequences of each trajectory which are similar to each other concerning their spatial characteristic. For the future we want to extend this approach by regarding rotations and forces. Another challenge would be to make the procedure invariant of scaling, by allowing deletions and insertions in our adaptation of the Smith-Waterman algorithm.

The algorithm described above tells us about the positions of matches within sequences, but does provide no information about the exact overlay of these matching regions. Therefore we will search for an optimal homogeneous transformation matrix whose application on one trajectory minimizes its displacement compared with the other one, within the matching region. We solve this problem by means of nonlinear programming (for a detailed introduction to this technique see [16]). Again, we only care about rotation and translation, while leaving it open to add scaling in a future version. The problem can be described as follows. As input data we have the same number of points $1 \dots m$ selected from the trajectories $A$ and $B$ by the Smith-Waterman algorithm. For data processing we

will use homogenous versions of the position vectors:

$$P_1^A = \begin{pmatrix} X_1^A \\ Y_1^A \\ Z_1^A \\ 1 \end{pmatrix} \cdots \begin{pmatrix} X_m^A \\ Y_m^A \\ Z_m^A \\ 1 \end{pmatrix};\ P_1^B = \begin{pmatrix} X_1^B \\ Y_1^B \\ Z_1^B \\ 1 \end{pmatrix} \cdots \begin{pmatrix} X_m^B \\ Y_m^B \\ Z_m^B \\ 1 \end{pmatrix};$$

We are searching for a homogenous transformation matrix, that has the following form:

$$H = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

For each matching pair of points, we now get a constraint describing the transformation:

$$\begin{aligned} HP_1^A + d_1^A &= P_1^B + d_1^B \\ &\vdots \\ HP_m^A + d_m^A &= P_m^B + d_m^B \end{aligned}$$

where $d_1^A \ldots d_m^A$ and $d_1^B \ldots d_m^B$ are homogeneous vectors enabling a residual distance between the corresponding points. Note that only one distance vector, e.g. on the left hand side of the equation, is not sufficient, because we want to minimize its absolute value, but functions (like $abs()$) cannot be used inside optimization problems. In addition, we have to guarantee that the resulting transformation matrix contains a valid rotation. For any rotation matrix it must hold true that every vector has unit length and each pair of vectors is perpendicularly aligned. These prerequisites can be formulated by dint of the dot product as nonlinear constraints. The resulting transformation matrix can be obtained from a nonlinear solver fed with the above formulation by minimizing the distance vectors. The result of such an optimization can be seen in the screenshot of our user interface for this application (fig. 12).

Once we have compared the first data points with the procedure described above, we have to find a mathematical description of the resulting primitive, (e.g. by analyzing its main axes and using them as a coordinate system to functionally approximate the position of the concomitant trihedron. With the help of this procedure we can build up a data base of primitives and can match newly acquired trajectories against it. If we compare many primitives with this method, we can determine the acceptable value facet for parameters describing the spatial position of our objects. Later these parameters can be adjusted within the determined range in order to react on certain situations with an appropriate skill.

### 3.3 High-Level Symbolic Representation

If we look at their underlying representations as curves, there is an obvious analogy between recongnizing two-dimensional handwriting (or printed text)

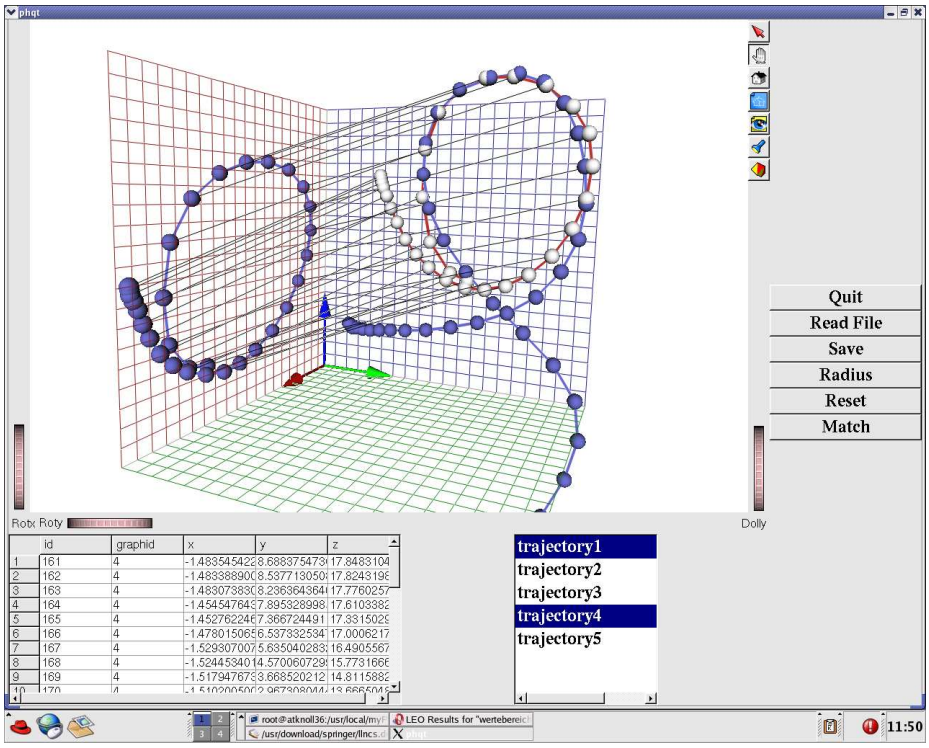


**Fig. 12.** Screenshot of the User Interface

and three-dimensional trajectories describing complex bimanual manipulation sequences. Therefore we rely on the established techniques developed in the Optical Character Recognition (OCR) domain, of course with the necessary adaptations to our domain, and suggest a hierarchical Hidden Markov Model (HMM) for identification and completion of manipulation sequences.

On the first level "*characters*" (a manipulation primitive) must be recognized. In OCR the digitally scanned pixel image from a character is analyzed and a couple of features are extracted like concavity, compactness, centroidal profile, number and position of holes, vertical/horizontal transitions. The correlation of these features to certain characters are trained by means of examples with HMMs. We could train in the same way manipulation primitives, with the difference, that the "*image*" of a "*character*" is composed of three curves, namely the spline and chaincode representations of positions, orientations and the forces occured. The features used by the correlation are those extracted from our two low-level symbolic representations (splines and 3D chaincodes): curvatures, torsions, Fourier descriptors and other features mentioned in section 2.2. Figure 13 shows a schematic "*character-level*" HMM, self-evidently there are as many HMMs as manipulation primitives.

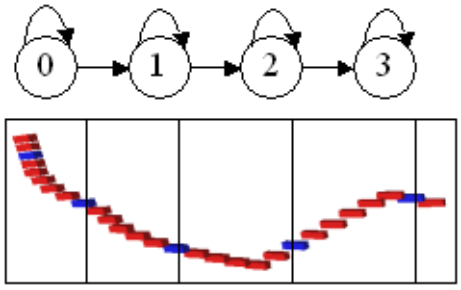


**Fig. 13.** HMM for recognizing "*characters*"

On the next level "*characters*" must be merged into "*words*". In the OCR this is done by word-level HMMs, which are trained using typical text passages, so that character transition probabilities are correlated to respective words. Then a dictionary query follows with the word one thinks it was recognized. We modify this approach, and make "*dictionary*" (see fig. 15) queries already after the first few recognized "*characters*". If there is an unambiguous match of the prefixes, then the word can be treated as recognized and the corresponding manipulation sequence can be completed on request. In the case of ambiquities further "*characters*" are necessary. Figure 14 shows the schematic "*word-level*", which is used to train whole manipulation sequences using HMMs.

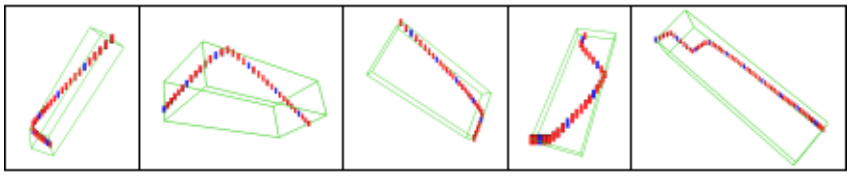

**Fig. 14.** A manipulation sequence "*word*"

### 3.4 On-line Completion

The on-line completion of partially recognized manipulation sequences was already hinted in the previous section. We want to mention here that in the case of a few number of ambiguous prefix matches the system could present them to the human operator, suggest the most probable and let him choose from the proposals. The decision taken by the human operator should also be included into the learning process. We didn't handle yet the case of no match at all, which is actually trivial: the appropiate manipulation sequence and his primitives are accepted as new entries.

We want to stress that learning does not take place in a dedicated learning phase, but during operation. Initially, the system has no knowledge about the application domain, but should learn it step by step by watching the operator. That means, at the beginning the operator has to do all the work alone, while in later phases the system can provide help in terms of semiautomatic execution of skills in certain situations. In order to meet all these requirements, we have to implement our solution in a way suited for real-time conditions.

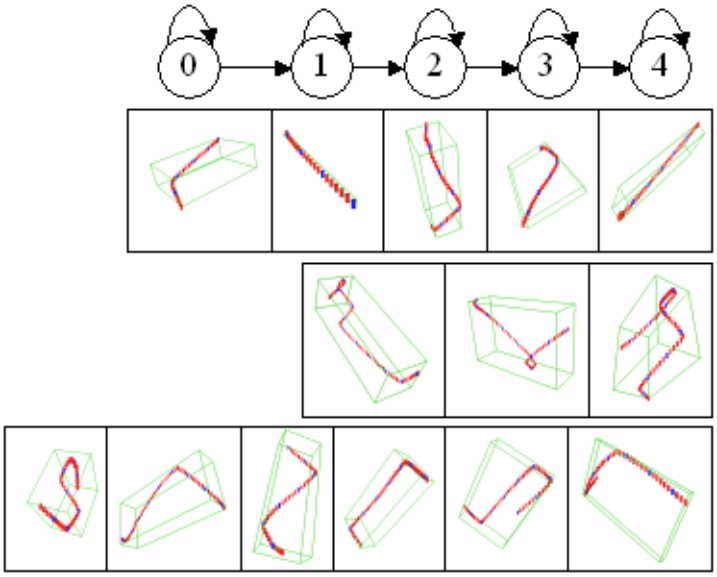


**Fig. 15.** A "*dictionary*" of manipulation sequences, each of them has its own HMM

## References

1. `http://www.kuka.de`
2. `http://www.intuitivesurgical.com`
3. G. S. Guthart, J. K. Salisbury: The *Intuitive*$^{TM}$ Telesurgery System: Overview and Application, IEEE ICRA, San Francisco, CA, April 2000.
4. `http://www.hbm.de`
5. M. Mitsuishi, S. Tomisaki, T. Yoshidome, H. Hashizume and K. Fujiwara: Telemicro-surgery system with intelligent user interface, IEEE ICRA, San Francisco, CA, April 2000.
6. H. Freeman: On the encoding of arbitrary geometric configurations, IRE Trans. on Electron. Comput., EC-10, 1961.
7. H. Freeman: Techniques for the digital computer analysis of chain-encoded arbitrary plane curves, Proc. Natl. Elect. Conf, 17, 1961.

8. H. Freeman: Computer processing of line drawing images. ACM Computing Surveys, 6, 1974.
9. E. Bribiesca: A chain code for representing 3D curves, Pattern Recognition, 33, 2000.
10. F.P. Kuhl, J. Weber, D. O'Connor and C.R. Giardina: Fourier series approximation of chain-encoded contours, Proc. Electro-Optics Laser 80 Conference and Exposition, Boston, MA, 1980.
11. F.P. Kuhl and C.R. Giardina: Elliptic Fourier features of closed contours, Technical Report, ARRADCOM, Dover, NJ, 1981.
12. Richard M. Voyles: Toward Gesture-Based Programming: Agent-Based Haptic Skill Acquisition and Interpretation PhD thesis Carnegie Mellon University, Pittsburgh (1997)
13. Michael Kaiser: Interaktive Akquisition elementarer Roboterfähigkeiten DISKI Vol. 153 Infix Verlag, St. Augustin, Germany 1997
14. Dan Gusfield: Algorithms on Strings, Trees and Sequences Cambridge University Press, New York (1997)
15. Volker Heun: Skript zur Vorlesung Algorithmische Bioinformatik Technische Universität München (2002)
16. Wayne L. Winston: Introduction to Mathematical Programming Applications and Algorithms Duxbury Press, Belmont (1995)