T U M

I N S T I T U T F Ü R I N F O R M A T I K

Application of Skill Transfer in Robotic Surgery

Istvan Nagy, Hermann Mayer, Alois Knoll

TUM-I0321
Dezember 03

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

Application of Skill Transfer in Robotic Surgery

Istvan Nagy, Hermann Mayer and 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

1 Introduction

The chair of Robotics and Embedded Systems at Technische Universität München has developed an experimental teleoperated robot system for minimally invasive surgery. In this scenario of robotic surgery, Skill Transfer from human demonstration to a machine, takes place in different subsequent steps. At first, user data is collected by means of two Phantom devices and the force sensors. With the help of these contrivances we can determine the position and orientation of the tool within the operation environment. If the tool gets in contact with the manipulated tissue or other objects in the environment, occurring forces are measured by strain gauges. By collecting this data over time, we get a complete description of the performed manipulation task. The collected raw data could be used for directly replaying the demonstrated task, since demonstration domain (user-controlled telemanipulator) and application domain (system-controlled telemanipulator) are identical in respect of the end effector. Note, that for many examples of Programming by Demonstration (PbD), domains are different, e.g. if a program for robot control is demonstrated with the user's hands. Although a direct replay would be possible, this is not the goal of our efforts. We rather want to generalize and contingently optimize the demonstrated task. Instead of simply collecting raw data, we want to get a packed symbolic representation of the task, which renders the system able to react to a-priory unknown situations. A concrete example for applying this capability is knot tying. Once the system has adopted this behavior from the user, it should be able to tie a knot in different orientations and detached from the spatial features of the observed demonstration. Therefore we have to extract task-relevant features from the demonstrations of the user. After one execution, the system is not able to distinguish important from irrelevant features of the task. For example it could be important to maintain a constant force while pulling a thread. On the other hand, the pursued trajectory might play no role. In order to differentiate such features, it is necessary to demonstrate the task many times. Thus, the first challenge is to extract significant features from the demonstrated tasks. Such a feature will be a temporally cohesive action, described by its trajectory and the occurring forces. We will call such a construct a primitive. Therefore a primitive is a function mapping time to certain rateable states of the environment, like

position, orientation and forces. If these quantities do not suffice to uniquely instantiate a primitive, we can add further features like distance, measured with a stereo camera system.
With a collection of primitives formed by this way, we should be able to sample every subsequent task demonstrated by the user. That means, we want to decompose an observed trajectory in some primitives. It is admissible that the sampling primitives overlap. If some parts of the task cannot be covered by any primitive, it is necessary to generate new primitives or alter existing ones. Therefore the act of primitive identification is never completed, but is a dynamic process, where primitives can be altered, generated, deleted or fused every time a task has been demonstrated.
If a task was observed sufficiently often, the system should be able to predict the next steps of an unfinished demonstration, which is subsequently performed by the user. Our goal is rendering the system able to semi-automatically complete tasks commenced by the user. Semi-automatic means in this sense, that the completion is not performed after recognizing the missing steps, but a simulation of one or more possible completions is displayed. The user can decide which of the proposals, if one is actually applicable, should be realized.

2 Methodology

Before we can start high level learning tasks, we have to deal with low-level data acquisition. Noise has to be filtered out and trajectories have to be represented in an appropriate way for further processing. One possibility is sampling the input stream at discrete points in time. Ways for representing the data are splines and chain codes. This preprocessed representation of data serves as an input for all further steps.
Learning in our system takes place in three entangled stages. The first stage is demonstrating the task sufficiently often, in order to enable the extraction of primitives. We want to implement the extraction by adopting on the one hand methods applied in Optical Character Recognition (OCR), and on the other hand methods from bioinformatics and text processing.
In order to extract primitives from multiple demonstrations, we have to look for similarities amongst them. Therefore we have to find matches between coordinate-based features in multidimensional space. If we restrict our endeavors to match 3D points of trajectories, we can find many approaches towards this issue. Most of them origin in text analysis and bioinformatics. We have the problem of discovering similar parts within acquired trajectories. This is very similar to challenges arising in bioinformatics, e.g. finding a protein fragment within a model of a virus. After identifying primitives, we have to recover them in subsequent demonstrations. In an optimal case, we should be able to sample the resulting trajectory with already known primitives. Since gaps and overlaps in this resampling cannot be avoided, we still must be able to create new primitives and generate new ones. Nevertheless, the main purpose of this stage is the refinement

of already known primitives. Since the knowledge about skills for performing certain tasks increases with each demonstration, we want to include this knowhow in the knowledge base of our system. In order to specifically include new insights extracted from current demonstrations, we need a learning strategy. We need to validate current instantiations of primitives (i.e. the concrete parameters of this primitive) against our knowledge base and alter it appropriately. This can be done by means of reinforcement learning.
In the last stage of learning, the system should be able to provide skills for performing different tasks. A skill is composed of one or usually more primitives. Our goal is, that the system is able to assist the user in performing recurrent task, but not by an application of a complete skill. It should rather recognize the commenced performance of a specific task and complete it by assuming the skill application of the user. That means at the beginning of a manipulation sequence, performed by the user, the system tries to recognize known primitives. These primitives are used to select a skill that is currently applied by the user. Triggered by such an recognition, the system automatically completes the current task within an internal simulation environment. It is possible that more than one skills are applicable in the current situation. After simulation, results are represented. Now, the user can decide, which simulated skill application should be realized by the system. In the following sections we will describe all mentioned procedures in detail.

3 Related Work

The complete scenario mentioned above was not mentioned by other authors in this compilation, but many related issues can be found in the corresponding literature. The main idea behind all developed methods in this context is to spare the user from dealing with programming details of the underlying system. Instead the system should learn from the user by observation. This paradigm is also called Programming by Demonstration and was addressed by several authors in the field of robotics (e.g. [KII94], [PJWI], [DH94] and [Kan94]). While early approaches mostly dealt with exactly replaying the user's demonstration (with high repeat accuracy), later contributions emphasize generalization and reuse of extracted primitives.
Skubic et al. [SV98] refer to primitives as sensorimotor skills. They have proposed a model that triggers the execution of skills by force-based events. Information is processed at a symbolic level. Therefore, with reservations, it is also possible to complete tasks under position and orientation variations. This is also our goal, but while Skubic et al. restrict their approach to force-based events and skills, we want to establish a model that also regards geometric aspects of the tasks and therefore can be employed for a broader variety of control tasks.
In order to classify our approach, we can use the terminology given by Kaiser et al. in [KFB$^+$]. In this sense, our method of generalizing multiple demonstration is a kind of "bottom-up skill acquisition". In contrast, "top-down skill acquisi-

tion" is understood as an implementation of a-priori domain knowledge. Since we want to decompose a known trajectory or a given task into previously acquired primitive, this part of our approach can be classified as top-down acquisition. Both methods can be supplied by "autonomous skill acquisition" that Kaiser describes as a means for skill refinement. An implementation of these theories can be found in [Kai96].

According to Hovland et al. [HSM96] a task can be represented ba a discrete state machine whose state is changed by events, e.g. performance of primitives. In the specified approach, the state machine is controlled by a Hidden Markov Model. We also want to use a modification of this approach for our learning tasks.

A combined top-down / bottom-up approach was proposed by Inamura et al. in [TIT01] and [TIN02]. They observed human motion with a motion capturing system and broke it up in larger segments with neural nets. These segments themselves were translated into a symbolic representation by means of Hidden Markov Models. Once an appropriate symbolic representation of a motion sequence have been found, it can be reconstructed by inversion of the previously described procedure. The resulting motion sequences can be used to replay a certain behavior with a robot.

Rosen et al. [JRS01] have employed a technique, which is very similar to that of Inamura. They have applied it for assessing the the training level of surgical residents. They also stated that their approach can be used as a performance measure for surgical robotic systems. The main idea of their work is to get a symbolic representation of certain demonstrations by means of Hidden Markov Models. By comparing corresponding HMMs of different groups (e.g. trainee vs. expert), one can derive a formal description of differences in performance. These experiments can be repeated from time to time in order to assess the advances of a trainee's skills.

In the approaches of Voyles [Voy97] and Friedrich et al. [HFD98] the demonstration is mapped on a predefined set of primitives. That means that an observed trajectory is decomposed in a symbolic representation consisting of various primitives. In our case we do not want to predefine a fixed set of primitives, but the system should learn this set from multiple demonstrations of the same task.

In the paper of Tominaga et al. [TTO^{+}00], primitives are called sub-skills. In their approach sub-skills are used to decompose user demonstrations to simplify analysis and trajectory generations. They only use geometric information derived from camera views, but do not measure forces. Like in the approach of Voyles, they decompose an observed demonstration in known primitives and reassemble them for robot control.

An interesting example of programming by demonstration was also performed by Chen et al. [CM98]. They showed that machine learning can best be understood, if user and machine have the same abilities. Since their system was not equipped with a camera system, they also wanted the user to demonstrate a pick in a hole task with closed eyes. Subsequently, the system calculates an optimal path from multiple demonstrations.

Slutski et al. have proposed in [LSE97] a functional decomposition of tasks into primitives, instead of a temporal one like many other authors. There idea was, that every human motion is a superposition of many independent sub-movements. For example the motion of the hand is composed of an upper-arm motion and a forearm motion. They have implemented this insight with a robot controller for telerobotic applications. Similar approaches of functional primitive decomposition can be found in [XZ00], [AM02] and [Mat01].

In the work of Nakamura et al. [YNM99] certain primitives, so-called reactive behaviors, are executed if the system is in a predefined stage. Complete skills are composed of several primitives in a way, that the resulting sensory output converges to the sensory output of the demonstration. This is an example of the adoption of reinforcement learning in skill transfer techniques.

Schaal et al. have proposed a complex mathematical model for primitive movements in [SSI03]. In their publication they proposed dynamic movement primitives based on nonlinear differential equations. They introduced a method for fast reinforcement learning based on these primitives.

While most authors have predefined their motion primitives, Fuentes et al. [FN97] have tried to learn the primitive themselves with the help of genetic programming (evolution strategy). They have used this approach for controlling the Utah/MIT hand. In our domain, motion primitives learned this way would only be useful, if we provide significant a-priory knowledge. Therefore we can use such an approach only for primitive refinement, but not for construction of completely new primitives.

Morrow et al. have also tried in [MK97] to provide a method for developing basic sensorimotor primitives. Their idea is based on the observation, that there are certain classes of relations between two manipulated objects (e.g. edge contact). A primitive serves for transforming one state into another. For example an alignment primitive is a transformation of a so-called free (unconstrained motion) state to an edge-surface or surface-surface contact. For a real-world implementation of such a primitive, force sensors are used. If contact occurs between two objects, (force is measures) the alignment-primitive is realized by minimizing occurring torques. This approach seems to work well for tasks comprising manipulation objects having a comparably simple shape (Morrow et al. have chosen a peg-in-hole task). Their methods cannot directly be transfered to our application domain without modifications, for we are dealing with non-uniform and deformable objects.

Another way of identifying primitives was proposed by Wang et al. in [WdS98]. They have used sudden changes in force measurement (contact information) to extract previously unknown primitives from demonstration.

Ogawara et al. have tried in [KOI03] to derive primitives only based on geometric features of demonstrations, while neglecting occurring forces and torques. Their approach is based on the observation of multiple demonstrations. For each demonstration the relationships between target object and manipulated object were recorded. In addition the user has to provide the system with a hint indicating essential interactions. By combining the knowledge of many demonstrations,

the essential primitives of those can be extracted.
Next, we want to give an overview on some methods applied in bioinformatics, that are related to our issues.
Fredriksson et al. introduced an approach for finding a protein fragment within a model of a virus (see [FU00]). They have generalized this example to finding a three dimensional pattern within a voxel model. The pattern can be arbitrarily rotated and translated. Their basic idea was to test all possible transformations in order to get the best match. They developed methods to significantly restrict the search domain. This approach works well for comparatively small patterns and models. We have made the experience that such approaches are not suitable for our domain, because it is not possible to perform realtime matching with large multidimensional models.

4 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.

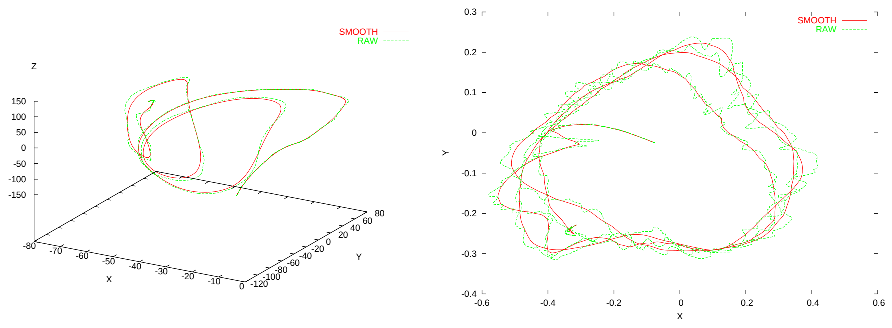


Fig. 1. Raw and smoothed data: (a) Positions; (b) Forces

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 1 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.

4.1 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.

4.2 Spline interpolation

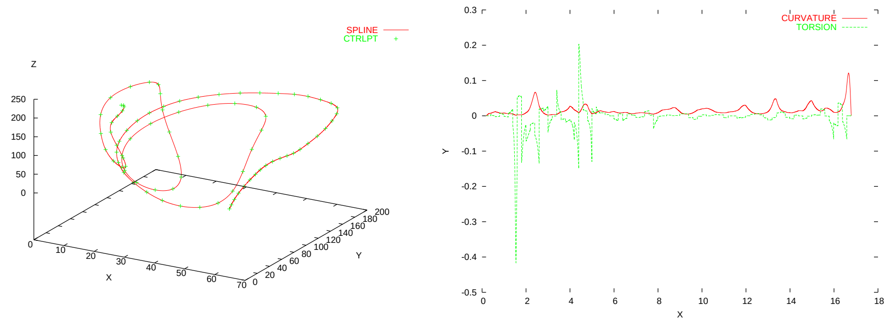


Fig. 2. (a) Spline interpolation; (b) Curvature and Torsion

For the spline interpolation natural cubic splines were used. Figure 2a shows an interpolating spline of the smoothed position trajectory from figure 1a. 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. 2b) 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.

4.3 3D Chaincoding

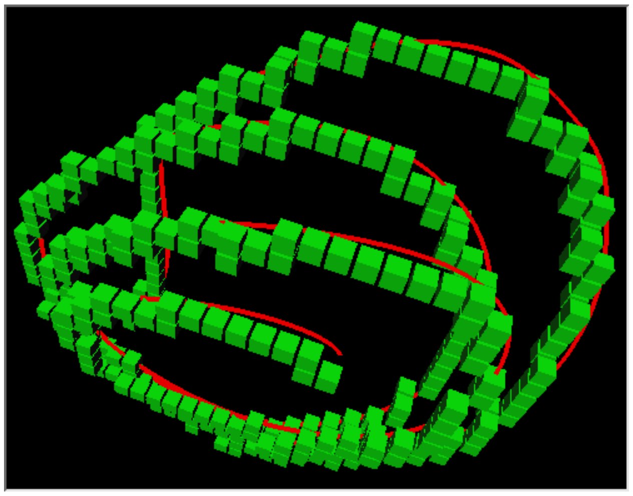


Fig. 3. 3D chaincoding

A method for representing digital curves using chain codes was first mentioned by Freeman in 1961 ([Fre61a], [Fre61b]). 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 ([Fre74]), 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 [Bri00] 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 3 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 [FKG80] and [KG81] 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 ψ-s curve, eccentricity, compactness, concavity, slope density function and shape numbers could also be evaluated.

5 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 [Voy97] or [Kai96]) 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.).

5.1 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.

Figure 4 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.

5.2 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

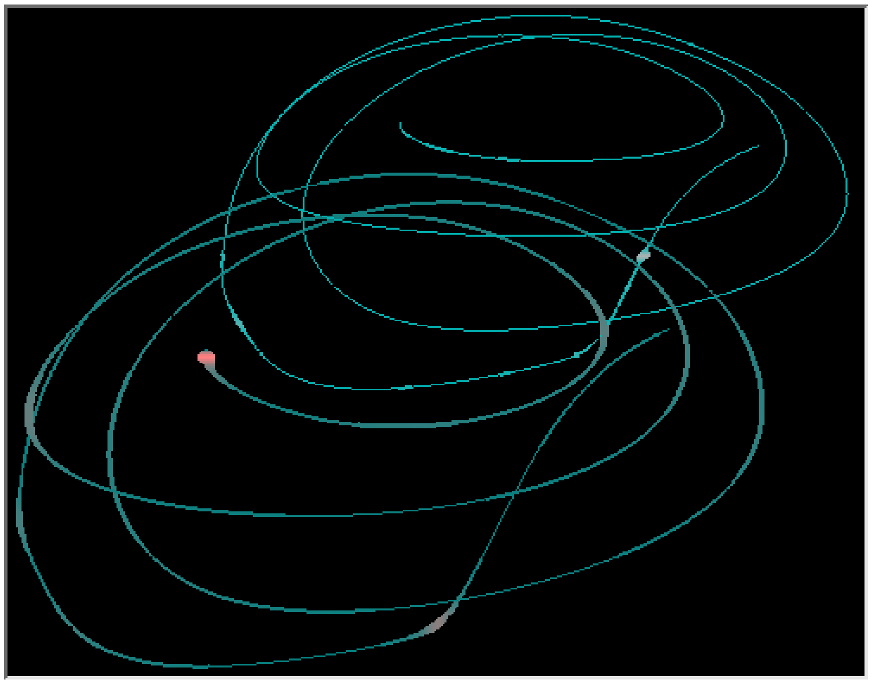


Fig. 4. Position trajectory spline augmented by curvature and torsion

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. 5).

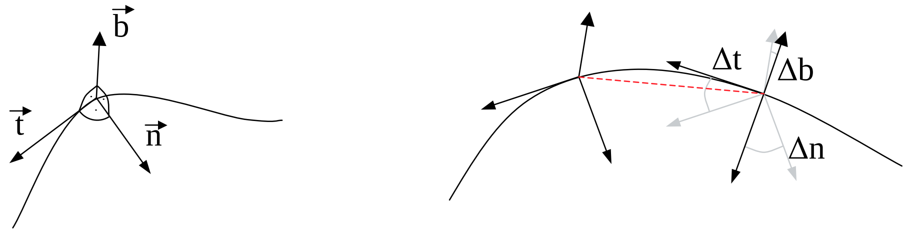


Fig. 5. Determining the difference between subsequent trihedrons

While each trihedron itself is expressed in terms of base coordinates, the difference 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. 6 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.

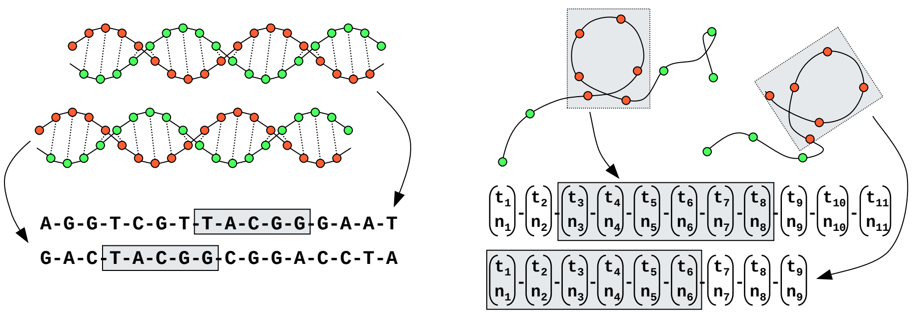

A-G-G-T-C-G-T-T-A-C-G-G-G-A-A-T
G-A-C-T-A-C-G-G-C-G-G-A-C-C-T-A
$t_1\,n_1$ - $t_2\,n_2$ - $t_3\,n_3$ - $t_4\,n_4$ - $t_5\,n_5$ - $t_6\,n_6$ - $t_7\,n_7$ - $t_8\,n_8$ - $t_9\,n_9$ - $t_{10}\,n_{10}$ - $t_{11}\,n_{11}$
$t_1\,n_1$ - $t_2\,n_2$ - $t_3\,n_3$ - $t_4\,n_4$ - $t_5\,n_5$ - $t_6\,n_6$ - $t_7\,n_7$ - $t_8\,n_8$ - $t_9\,n_9$

Fig. 6. Analogy between Bioinformatics and Robotics

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. 6). Next we select an appropriate algorithm 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 [Gus97] or [Heu02]. 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 7, a local alignment can be constructed by the following operations:

1. assume a deletion in comparison with the other string (vertical movement in fig. 7)
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 7 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. 7). 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 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

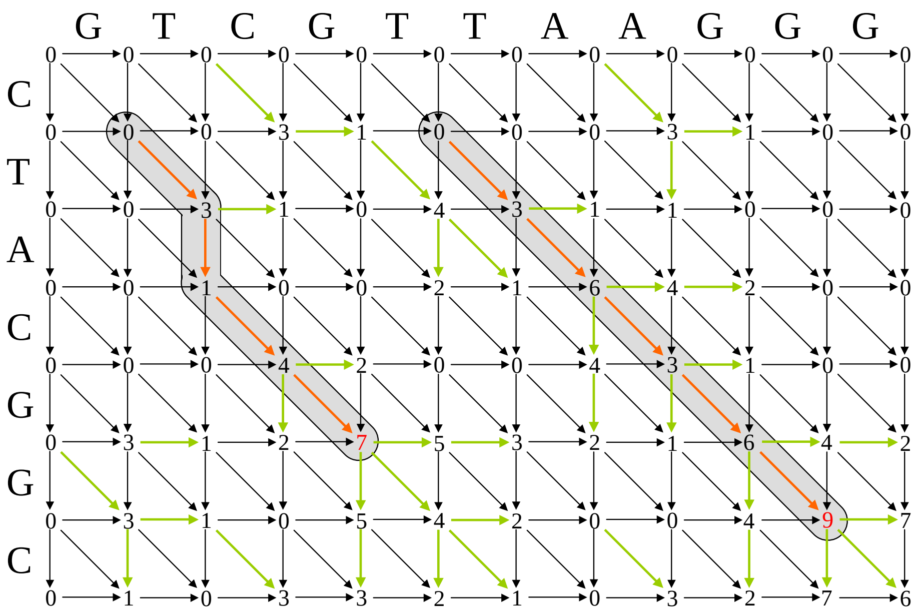


Fig. 7. Example for a local alignment search

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 [Win95]). 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:

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

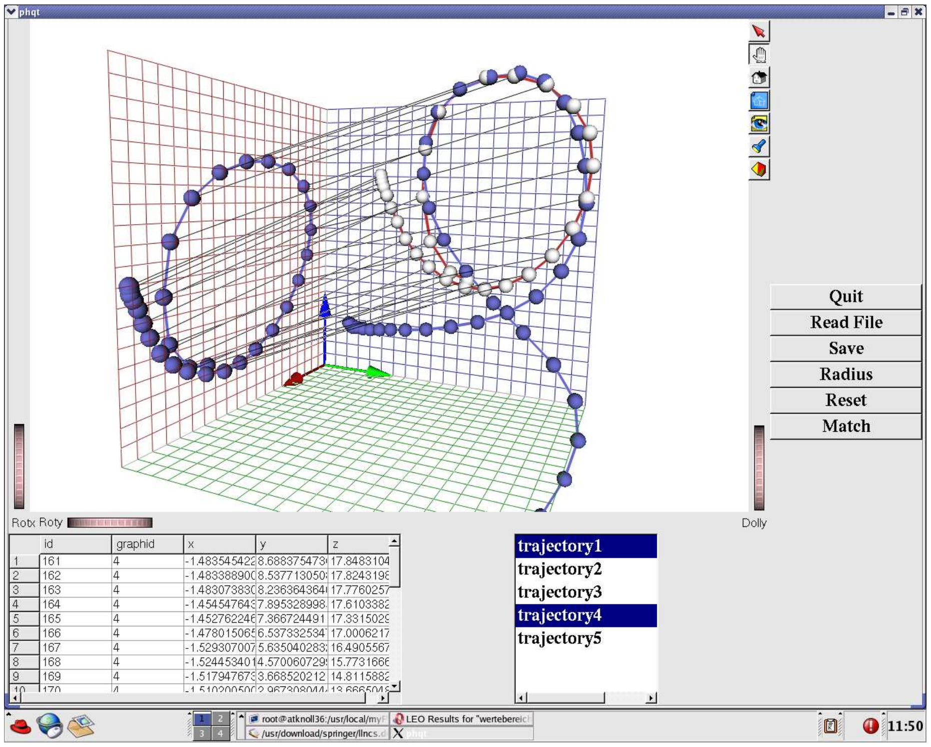
We are searching for a homogenous transformation matrix, that has the following form:

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

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

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

where $d_1^A \dots d_m^A$ and $d_1^B \dots d_m^B$ 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. 8).


phqt
Quit
Read File
Save
Radius
Reset
Match
Rotx Roty
Dolly
root@atknoll36:/usr/local/myF
/usr/download/springer/llncs.d
LEO Results for "wertebereic
11:50

	id	graphid	x	y	z
1	161	4	-1.483545422	8.688375473	17.8483104
2	162	4	-1.483388900	8.537713050	17.8243198
3	163	4	-1.483073830	8.236364364	17.7760257
4	164	4	-1.454547643	7.895328998	17.6103382
5	165	4	-1.452762246	7.366724491	17.3315029
6	166	4	-1.478015065	6.537332534	17.0006217
7	167	4	-1.529307007	5.635040283	16.4905567
8	168	4	-1.524453401	4.570060729	15.7731666
9	169	4	-1.517947673	3.668520212	14.8115882
10	170	4	-1.510200500	2.967308044	13.6665048

trajectory1
trajectory2
trajectory3
trajectory4
trajectory5

Fig. 8. Screenshot of the User Interface

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.

5.3 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) 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 4.3. Figure 9 shows a schematic "*character-level*" HMM, self-evidently there are as many HMMs as manipulation primitives.

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. 11) 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 10 shows the schematic "*word-level*", which is used to train whole manipulation sequences using HMMs.

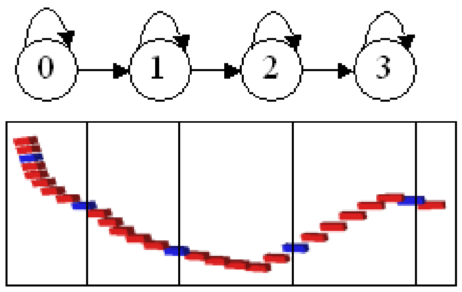


Fig. 9. HMM for recognizing ”*characters*”

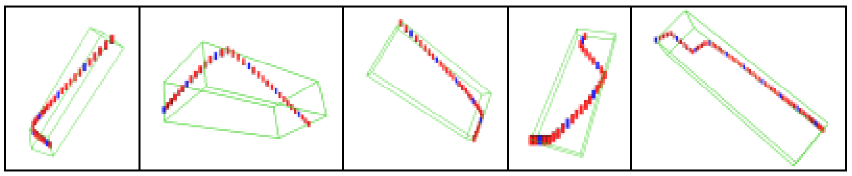


Fig. 10. A manipulation sequence ”*word*”

5.4 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.

References

[AM02] R. Amit and Maja J. Mataric. Parametric primitives for motor representation and control. *IEEE International Conference on Robotics and Automation*, pages 863–868, 2002.

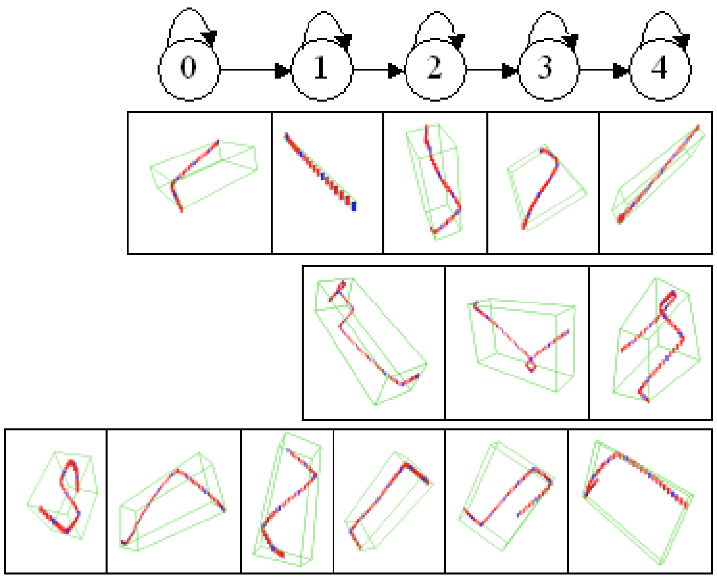


Fig. 11. A "*dictionary*" of manipulation sequences, each of them has its own HMM

[Bri00] E. Bribiesca. A chain code for representing 3d curves. *Pattern Recognition, 33*, 2000.

[CM98] J. Chen and Brenan McCarragher. Robot programming by demonstration - selecting optimal event paths. *IEEE International Conference on Robotics and Automation*, pages 518 – 523, 1998.

[DH94] J. Demiris and G. Hayes. A robot controller using learning by imitation, 1994.

[FKG80] D. O'Connor F.P. Kuhl, J. Weber and C.R. Giardina. Fourier series approximation of chain-encoded contours. *Proc. Electro-Optics Laser 80 Conference and Exposition, Boston, MA*, 1980.

[FN97] O. Fuentes and R. C. Nelson. Learning dextrous manipulation skills using the evolution strategy. *IEEE International Conference on Robotics and Automation*, pages 501 – 506, 1997.

[Fre61a] H. Freeman. On the encoding of arbitrary geometric configurations. *IRE Trans. on Electron. Comput., EC-10*, 1961.

[Fre61b] H. Freeman. Techniques for the digital computer analysis of chain-encoded arbitrary plane curves. *Proc. Natl. Elect. Conf, 17*, 1961.

[Fre74] H. Freeman. *Computer processing of line drawing images.* ACM Computing Surveys, 1974.

[FU00] Kimmo Fredriksson and Esko Ukkonen. Combinatorial methods for approximate pattern matching under rotations and translations in 3d arrays. In *String Processing and Information Retrieval*, pages 96–104, 2000.

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, New York, 1997.

[Heu02] Volker Heun. *Skript zur Vorlesung Algorithmische Bioinformatik.* Technische Universitaet Muenchen, 2002.

[HFD98] J. Holle H. Friedrich and R. Dillmann. Interactive generation of flexible robot programs. *IEEE International Conference on Robotics and Automation*, pages 538 – 543, 1998.

[HSM96] G. Hovland, P. Sikka, and B. McCarragher. Skill acquisition from human demonstration using a hidden markov model. *IEEE International Conference on Robotics and Automation*, pages 2706–2711, 1996.

[JRS01] Blake Hannaford Jakob Rosen, Missimiliano Solazzo and Mika Sinanan. Objective laparoscopic skills assessments of surgical residents using hidden markovmodels based on haptic information and tool/tissue interactions, 2001.

[Kai96] Michael Kaiser. Interaktive akquisition elementarer roboterfaehigkeiten, 1996.

[Kan94] S. Kang. Robot instruction by human demonstration, 1994.

[KFB$^+$] Michael Kaiser, Holger Friedrich, Rob Buckingham, Koorosh Khodabandehloo, and Simon Tomlinson. Towards a general measure of skill for learning robots.

[KG81] F.P. Kuhl and C.R. Giardina. Elliptic fourier features of closed contours. *Technical Report, ARRADCOM, Dover, NJ*, 1981.

[KII94] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *T-RA*, 10:799–822, 1994.

[KOI03] Hiroshi Kimura Koichi Ogawara, Jun Takamatsu and Katsushi Ikeuchi. Estimation of essentinal interactions from multiple demonstrations. *IEEE International Conference on Robotics and Automation*, pages 3893–3898, 2003.

[LSE97] I. Gurevich L. Slutski and Y. Edan. Analysis and design of telerobot control based on human motor behavior study. *IEEE International Conference on Robotics and Automation*, pages 1016 – 1021, 1997.

[Mat01] Maja J. Mataric. Visio-motor primitives as a basis for learning by imitation, 2001.

[MK97] J. D. Morrow and P. K. Khosla. Manipulation task primitives for composing robot skills. *IEEE International Conference on Robotics and Automation*, pages 3354 – 3359, 1997.

[PJWI] George V Paul, Yunde Jiar, Mark D Wheeler, and Katsushi Ikeuchi. Modelling human assembly actions from observation.

[SSI03] Jun Nakanishi Stefan Schaal, Jan Peters and Auke Ijspeert. Learning movement primitives. *International Symposium on Robotics Research (ISRR2003)*, pages 863–868, 2003.

[SV98] M. Skubic and R. A. Volz. Learning force-based assembly skills from human demonstration for execution in unstructured environments. *IEEE International Conference on Robotics and Automation*, pages 1281–1288, 1998.

[TIN02] Iwaki Toshima Tetsunari Inamura and Yoshihiko Nakamura. Acquisition and embodiment of motion elements in closed mimesis loop. *IEEE International Conference on Robotics and Automation*, pages 1539–1544, 2002.

[TIT01] Hideaki Ezaki Tetsunari Inamura, Yoshihiko Nakamura and Iwaki Toshima. Imitation and primitive symbol acquisition of humanoids by the integrated mimesis loop. *IEEE International Conference on Robotics and Automation*, pages 4208–4213, 2001.

[TTO$^+$00] H. Tominaga, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi. Symbolic representation of trajectories for skill generation. *IEEE International Conference on Robotics and Automation*, pages 4077 – 4082, 2000.

[Voy97] R. Voyles. Toward gesture-based programming: Agentbased haptic skill acquisition and interpretation, 1997.

[WdS98] Q. Wang and J. de Schutter. Towards real-time robot programming by hunman demonstration for 6d force controlled actions. *IEEE International Conference on Robotics and Automation*, pages 2256 – 2261, 1998.

[Win95] Wayne L. Winston. *Introduction to Mathematical Programming, Applications and Algorithms*. Duxbury Press, Belmont, 1995.

[XZ00] L. Xu and Y. F. Zheng. Real-time motion planning for personal robots using primitive motions. *IEEE International Conference on Robotics and Automation*, pages 1414 – 1419, 2000.

[YNM99] T. Yamazaki Y. Nakamura and N. Mizushima. Synthesis, learning and abstraction of skills through parameterized smooth map from sensors. *IEEE International Conference on Robotics and Automation*, pages 2398 – 2405, 1999.