**ABSTRACT**

In the context of service robotics very often no model of the environment exists. The actual situation can partially be recognized by sensors as a snapshot, which is then used for planning an action, the optimal trajectory, however, e.g. in manipulation tasks, to reach a desired destination is often very difficult to compute, because not all aspects were catched by the sensors. Therefore, systems with rule based behavior can be used to autonomously plan the motion online using mainly tactile sensor information. For repeating the same or a similar task, it would be nice to have motion sequences allowing much faster motions with less collisions. The problem we adress in this paper is the transition from rule based behavior towards motion sequences without loosing the flexibility of the system, which is necessary to adapt a motion sequence to the actual situation and to deal with unforeseen events during execution.

**1 INTRODUCTION**

During the last few years increasing attention is paid to topics concerning service robotics. One of the key problems in this area is the complexity of the tasks to be fulfilled. Because it is not possible to have always an engineer nearby to interact, it is inevitable to have systems with a high level of autonomy. This is even more necessary because of a lack of precise models of the environment which leads to the problem of state uncertainties. Thus it is expected that those systems are able to deal with variations of a task and to react flexible on unexpected situations still trying to reach the goal. Therefore sensor information must be used to characterize situations.

**Motions in Only Partially Known Environment**

While the initial state of the environment can partially be recognized as a snapshot, which is then used for planning an action, the optimal trajectory, however, e.g. in manipulation tasks, to reach a desired destination is often very difficult to compute, because not all aspects were catched by the sensors and a geometric model of the environment is not available.

One solution could be to try to build a detailed model, but this needs a variety of sensors, takes a lot of time, and is only valid for a very short time, because in most cases the environment is changing and not static. Another way is to specify the desired behavior of the system so that it is able to plan the trajectory autonomously during the motion. The system we use for the disassembly problem - discussed more detailed in the next section - obtains this ability by superposing local evading movements, according to heuristic rules, to a global direction towards the destination, and thus can reach the goal itself (see e.g. [11], [12]).

On the other hand, especially for manipulation, a system can use the same type of motion for different tasks, if it is the same basic problem. Thus it would be nice to have already a motion sequence allowing a faster performance with less collisions than a motion, which during execution is based on tactile information only. In literature different approaches for learning a given temporal sequence of patterns, also e.g. motor commands for a robot system, can be found (see e.g. [2], [8], [1], [5]).

The question arising is, how can we achieve a transition from the rule based behavior, which helps to solve the problem of the unknown trajectory and keeps the system flexible with regard to unforeseen situations, towards the motion sequences, which allow increased speed with less collisions, as soon as the same type of motion is required again, without loosing the flexibility of the system?

**Previous Work**

For solving open questions in this field of research it proved to be quite useful to have a look at biological systems, especially humans, not to copy and rebuild them but to imitate some of their abilities in a suitable way. Humans have this ability. If they are new to a problem they use their over all experience, which can be seen as heuristic rules giving them a basic idea of how to solve the problem. As soon as they aquired some knowledge about the new task, they execute the motions much faster by using motion sequences, or more generally motor programs\(^1\). Considering a model of the human motor system as e.g. proposed in [4], the rule based behavior is for motions, which are started and performed consciously. The motor programs are used for motions which are started consciously but performed unconsciously, comparable to stepwise feed-forward control. The transition is realized

---

\(^1\)Here a motion sequence can be seen as a single instance of a motor programme which groups sequences of the same type together, still allowing to adapt the concrete instance to the actual situation e.g. by geometric distortion or necessary online corrections (for the idea of motor programs see also e.g. [9]).
by structuring the information acquired during repeating the task, e.g. like a tennis player learning a forehand drive.

For learning structured knowledge Hidden Markov Models (HMMs) have been used successfully e.g. for speech recognition (e.g. [7]), mobile robot path planning (e.g. [14]) and also for skill learning problems (e.g. [13]). Especially for manipulation tasks we developed an architecture using a discrete HMM to realize the transition from rule based behavior to motion sequences (see [6]). Based on a number of successful motions a suitable motion sequence describing the required trajectory for that task can be modelled by a HMM, and then be used to generate motor commands for the robot system.

Thus it is possible first to adapt the motion by geometric distortion to the actual situation as caught by the sensors (e.g. vision) and then execute it with higher speed but less collisions as the tactile rule based movement.

However, the problem still left is the loss of the flexibility of the system. Of course a supervising level can stop the motion if unforeseen situations occur and switch back to the rule based behavior, but very often some online adaptations of the motion sequence would be sufficient to continue. Furthermore, already after the first successful motion, the system found a trajectory which could be used as an initial solution for the problem, which can then be refined during further execution, so that the advantage of increased performance is directly available from the second motion onward.

**Aim of this Work**

In this paper we present an approach to achieve both aims, to have a system, which is able to start with rule based behavior for tasks where the trajectory leading to the desired destination is unknown, and then directly builds up a generic motion sequence (motion program), which is further refined by incremental learning based on the information acquired during repeated execution, and on the other hand keeps it’s flexibility allowing to continue a motion sequence even if online modifications are necessary due to unforeseen events or due to greater variation from the learned problem than expected.

Next the disassembly problem is described, which we use as an example application for our system. After the architecture and idea of our new approach is outlined, we present experimental results achieved with the real robot system in our laboratory. The paper ends with a summary and conclusions and an outlook on future work.

**2 THE DISASSEMBLY PROBLEM**

Automated disassembly is a typical manipulation task where a robot system has to perform a motion in an only partially known environment. Originally we started to work on the disassembly of end-of-life products of electronic devices. Due to the already very high, nevertheless still increasing number of such products, non-destructive disassembly and re-use of valuable modules is an environmentally sensible way to face this problem. But also for economical reasons because of the resulting quantity of devices to be recycled, increasing automation seems to be necessary.

However, non-destructive disassembly is also required for inspection tasks, if first another object has to be removed before the interesting one can be examined, or also for automated replacement of parts, or for repair of technical devices, or also removal of objects. This is especially interesting in areas difficult to reach by humans as e.g. space or under water applications, or also nuclear power plants.

Figure 1a) shows e.g. a camcorder with the lens module on the left side of the casing being the most interesting part for reuse, because the lenses are valuable and normally not damaged during use of the device. Figure 1b) illustrates a typical sequence of some situations during the disassembly of the lens module in a simplified schematic representation, beginning with the initial state. There the module is inside with geometric constraints of its movability caused by neighboring modules as the electronics and also by the casing, which furthermore has an undercut so that it is not possible to take out the lens module in a straight motion. For those products normally no model exists, so an autonomous system has to rely only on the information it can acquire by its own sensors. After localizing the object and grasping at a suitable grasp point the object is removed by a rule based strategy with heuristic rules stored in a fuzzy controller. The only information given to the system is the direction of a global disassembly motion, which is just the reverse direction of approaching the object during grasping. By superposing local evading movements to the global direction, according to the heuristic rules, as a reaction to forces and/or torques measured online, the system can reach the goal itself (see e.g. [11], [12]). Figure 2 shows the control structure used to realize the rule based behavior.

Having a look at the schematic representation of figure 1b) it is easy to imagine that the same situation and type of motion is also necessary for complete different fields of applications, e.g. a service robot at home as support for elderly or handicapped persons, when opening the drawer of a cupboard to get out something as e.g. a donut box. Thus non-destructive disassembly operations occurs in a wide range of applications, often with somehow similar basic types of motions.

**3 INCREMENTAL LEARNING AND REFINEMENT OF MOTION SEQUENCES**

In the approach we presented in [6] we used a number of successful motions to train a discrete HMM which then is used to derive a suitable motor command sequence for the robot. If, however, even after one successful motion already a sequence can be generated, which provides advantages compared to the rule based behavior, an incremental process for learning of a motion sequence should be preferred. In that case also the use of the HMM should be thought over, because it’s statistical features are no longer needed. Thus only a suitable representation of succeeding states describing the trajectory is necessary. In [10] force-based qualitative states for a finite state machine are suggested. Disassembly experiments, however, proved that it is

---

2The disassembly is supposed to be successfully completed, if the object is moved to a certain distance outside the casing.
impossible to define characteristic and unique states along the trajectory based on force/torque measurements, because of the high number of collisions with other objects, due to the very limited clearance inside the casing, especially in the beginning of the training, when the trajectory to learn is unknown. Therefore we keep the definition of the discrete state vectors as in [6], so state \( s_i \) is defined as a combination of the pose \( p_i \) at time \( t_i \) and \( t_i \) itself:

\[
\begin{align*}
    p_i &= (p_{xi}, p_{yi}, p_{zi}, p_{dx}, p_{dy}, p_{dz})^T \\
    \Rightarrow s_i &= (p_{i}^T, t_i)^T
\end{align*}
\]

(1)

(2)

Because we want a discrete number of states, vector quantization by a Growing Neural Gas (GNG) network\(^3\) is used to cluster the time and position sensor data of the successful motion. The sequence of the states results of the measured data classified by the GNG network. With a slight internal modification of the control structure of figure 2 this state sequence can be used as a motor command sequence. Instead of using the global motion direction \( \mu(k) \) the strategy level uses the state sequence. At the start of the motion and each time switching from one state to the next, the direction from the actual pose to the pose encoded in the next state is computed, and the distance is scaled down to get a desired velocity, suitable as robot command in each sampling unit\(^4\). Switching from state \( s_i \) to \( s_{i+1} \), being the next state to reach, is done if the system passes a hyper-plane \( h_s \) defined by:

\[
\begin{align*}
    h_{si} & : (\xi - p_{xi})\mu = 0 \\
    \mu & \in h_{ip} = p_{ti} + \lambda u_k + \mu p_{ti} \\
    0 &= \mu(u_k + \mu) \\
    u_k &= \frac{p_{t_{i+1}} - p_{ti}}{\|p_{t_{i+1}} - p_{ti}\|}
\end{align*}
\]

(3)

(4)

(5)

(6)

\(^3\)For details about Growing Neural Gas networks see e.g. [3].

\(^4\)This scaling modifies the timing of the sequence, thus the time \( t_k \) of a state \( s_k \) is no longer necessarily identical with the real time in the motion, but can be seen as a point on a virtual time base (cp. [5]). Furthermore, an individual scaling for each component of the velocity allows an overall geometric distortion of the motion sequence if desired, e.g. if the analysis of the initial situation suggests this.
The calculated velocity is then kept as a constant motor command input to the non-linear mapping performed by the fuzzy controller of the rule based architecture. As long as no modifications of the motion sequence are necessary, due to collisions, the motor commands are passed through to the robot. As soon as collisions occur, because the motion trajectory is not yet optimal, or because of an unforeseen situation or changes in the environment, the system is able to autonomously adapt the motion trajectory.5. This way we have a very easy possibility to generate and use motion sequences, to improve speed and the number of collisions, and still preserve the flexibility of the system. If the motion sequence fails, i.e. all states would “used up” without reaching the goal, the system assumes that the problem is too different, so that the chosen sequence does not fit. It then totally switches back to the rule based behavior and tries to find a new trajectory.

To further improve the performance force/torque information can be used to enhance the motion sequence. Therefore, vector quantization by GNG networks is also done for the force/torque sensor data. For each state the most probable external impact is determined and according to this the poses of the states of the motion trajectory are modified:

\[
\begin{align*}
  K_M &= \text{diag}(k_{MN}), k_{MN} > 0, \ n = 0 \ldots 5 \\
  f_l &= (f_{x\ell t}, f_{y\ell t}, f_{z\ell t}, \tau_{x\ell t}, \tau_{y\ell t}, \tau_{z\ell t})^T
\end{align*}
\] (10)

To avoid a too much bulged motion sequence on the one hand, and on the other hand because - especially at beginning of learning - the sequence is not yet optimal, which does not cause problems due to the possibility of interactions by the underlying feedback control by the rule based behavior, as well as to get a smooth motion trajectory, after the modification of the states an additional low pass filtering of the state sequence is done. Therefore each component of the pose of each state is filtered, still considering the forces in that direction not to revoke the effects of the first modification:

\[
\begin{align*}
  p_{x\ell} &= K_{G1} p_{x\ell} + \sum_{k=1}^{m} K_{G2} p_{x(k+1)\ell} \\
  K_{G1} &= \text{diag}(\omega_n) \\
  K_{G2} &= \text{diag}(1) \\
  \omega_n &= k_{G2, n}|f_{n, \ell t}| > 0, \ n = 0 \ldots 5
\end{align*}
\] (11-13)

This final state sequence is given to the strategy level, which uses it to generate the motor commands as described above. After the motion the new sensor data is used to calculate a new sequence for refining the motion and stepwise improve the behavior by incremental learning.

For the case, that necessary evading movements prevent the system ever to pass the hyper-plane, a second condition for switching to the next state is a scaled difference of \(t_{\ell t} - t_{\ell t-1}\). As soon as this interval is exceeded the next state \(s_{\ell t+1}\) is to reach so that the motion goes on in a slightly new direction.

\[
\nu_2 = \frac{p_{x\ell+1} - p_{x\ell}}{|p_{x\ell+1} - p_{x\ell}|}
\] (7)

The experiments were done with a six axes puma type robot at our laboratory. The discussion will focus on the disassembly of the lens module out of the camcorder, being a motion in a real and complex environment. Suitable and very robust parameters were easily found, and we could use identical parameters allow:

\[
k_{MN} = k_{2, n} = 2, \ n = 0 \ldots 5
\] (15)

Figure 3 shows the measurement data of the main components of the motion performed by only using the rule based behavior. Because the system has no idea of the real trajectory to get out, a lot of collisions occur leading to oscillations in the movement and thus to lost time and stress of the disassembly object and the environment.

In comparison figure 4 shows the measurement data after three repetitions. It can easily be seen that the computed state sequence (illustrated in black, each asterisk means a state) is already a good and smooth approximation of the optimal trajectory. Thus there are now less and especially less harsh collisions. The performed motion (illustrated in grey, each cross indicates a switch to the next state) is only in some areas modified, done autonomously by the system. Furthermore, the time needed for the disassembly operation decreased from about 30 seconds to less than 20 seconds.

We defined a simple quality function, which should be minimal, to judge the overall behavior with emphasis on the two most interesting points, the total disassembly time \(\Theta\) and the accumulated absolute force/torque measurements \(F_{f\ell}\) as characterization for the collisions:

\[
J = \frac{1}{2}\left(\frac{\Theta}{\Theta_{RB}} + \frac{F_{f\ell}}{F_{f\ell, RB}}\right)
\] (16)

\[
F_{f\ell} = \sum_{n=1}^{6} \frac{1}{\Theta} \sum |f_{n, \ell t}|
\] (17)

The values of the actual motion are standardized to the values of the original rule based motion, indicated by the index \(RB\), thus the original motion is judged with the value \(J = 1\). Figure 5 shows how the behavior is significantly improved during the first repetitions. Then the values oscillate at about \(J \approx 0.7\).

5 SUMMARY AND CONCLUSIONS

We presented an approach to realize the transition from rule based behavior towards motion sequences for motor control of a flexible robot system by using incremental learning. In contrast to other methods our system is able to start with rule based behavior for tasks where the trajectory leading to the desired destination is unknown, and then directly builds up a generic motion sequence, which then after each repeating execution can further be refined by incremental learning based on the information acquired, and also keeps it’s flexibility allowing to continue a motion sequence even if online modifications are necessary due to unforeseen events or due to greater variation from the learned problem than expected.

As a result the system can improve its behavior already after the first motion and can then further improve its behavior.
Figure 3: These Diagramms show the measurement data of the main components of the motion, performed using the rule based behavior only. (Note: The data is plotted with respect to task coordinate frame $T$ at start of the motion, as in figure 1)

Figure 5: The diagram shows the values of the quality function in the middle, above is the standardized accumulated force/torque value, below is the standardized total time needed for the whole disassembly operation.

by collecting further experience and so incrementally learns about the problem.

For future work we think about an architecture, where the motion is no longer represented as a sequence of discrete states but directly as a continuous motion, e.g. in an interpolating memory, with also including the collision information. This should then be the basis for generating a motion command sequence, which can be given to the strategy level of the control structure of figure 2, again to keep the flexibility of the system while in parallel incrementally optimize the behavior.

REFERENCES


Figure 4: These diagrams show the measurement data of the main components of the motion, performed using the motion sequence after three repetitions. (Note: The data is plotted with respect to task coordinate frame $T$ at start of the motion, as in figure 1.)


