COMBINING MANUAL HAPTIC PATH PLANNING OF INDUSTRIAL ROBOTS WITH AUTOMATIC PATH SMOOTHING

Heinz Wörn, Björn Hein, Detlef Mages
Institute for Process Control and Robotics (IPR), Universität Karlsruhe (TH), Engler-Bunte-Ring 8, 76131 Karlsruhe, Germany,

Berend Denkena, Rene Apitz, Pawel Kowalski
Institute of Production Engineering and Machine Tools (IFW), Universität Hannover, Schoenebecker Allee 2, 30823 Garbsen, Germany,

Niels Reimer
Tecnomatix GmbH, Richard-Reitzner-Allee 8, 85540 Haar (München), Germany

Keywords: robot programming, path smoothing, haptic manipulation, robot simulation, industrial robotics.

Abstract: Nowadays, industrial robots are preferably programmed offline, i.e. without interference with the real cell and running production processes. Usually a simulation tool is used to manually define individual locations and to check the created raw path for possible collisions. Within this paper an approach is presented, that combines a haptic input device by means of automated path smoothing. The quality of the generated path can be significantly improved by subsequent automatic filtering. Removing redundant locations or modifying intermediate ones increases the smoothness of the path. The semi-automatic programming paradigm with haptic interaction is expected to lead to an improved workflow for robot offline programming.

1 INTRODUCTION

Today, offline robot paths are usually defined manually, which is a time-consuming and error-prone process, in particular with regard to the fact that industrial robots have up to six degrees of freedom. This paper discusses an integration of means of virtual reality and automated path smoothing.

Figure 1: Coordination of haptics and the application

In robot simulation systems paths are traditionally created with a mouse. First the robot is moved to the desired start. This can normally be done by moving the axes of the robot individually or by moving the tool center point. Often a robot can reach one location with different poses so the user must select one via a configuration dialog. When the robot is in the right position, the user adds a new point and places it at the desired position. After that, the created path is visually checked for collision-freeness. This cycle is repeated until a collision-free path from start to goal is generated. Due to the high number of iteration steps, this process is time-consuming and very monotone (repeating the same steps over and over) for the user.

2 HAPTIC PATH GENERATION

Virtual reality covers perception by all human senses. One of these senses is the sense of touch, i.e. haptics (Burdea, 1996). Nowadays, the most popular fields of application for this science are medicine and engineering. Within the field of medical sciences, haptics has been successfully used for training and simulation of surgeries (Cakmak and Kühnapel, 2000). Within engineering applications, haptic devices have been first used for modeling complex free-form objects, where virtual clay is used for designing (Fujiiki and Aoyama, 1999). Haptic devices can also be used for programming machine tools (Tönshoff et al., 2001). Efforts have also been made to interact with robots via haptic devices. These approaches, however, focus mainly on direct manipulation (Kesavadas
The path smoothing algorithm presented here is based on the method described in (Berchtold and Glavina, 1994) and tested in (Hein, 2003). Its input is a path that is either generated by an automatic algorithm or manually drafted e.g. by a haptic device. A path consists usually of so called point to point (PTP) movements. PTP means that all axes of the robot are moved simultaneously along a straight line in the configuration space of the robot. Start and end location of a PTP movement are reached simultaneously. The algorithm tries to erase points that are redundant in two senses. First, collinear segments can be reduced (remove the intermediate points) to one segment, see section 3.1.1. Secondly, vertices that can safely be deleted without harming the collision-freeness of the path are considered redundant, see section 3.1.2.

The algorithm smooths the path by inserting and deleting vertices at promising locations. In the following text the notion “sharp vertex” is used, which is a vertex that has a small angle defined by its adjacent segments. The input data for the algorithm is a (rough) path in configuration space that has to be collision-free. All robot poses along the path in configuration space must result in a collision-free situation in work space. (Mages et al., 2004; Schwarzer et al., 2004)

### 3.1 Components of the algorithm

#### 3.1.1 Linear redundant points

Linear redundant points can be identified by the angle between two neighboring segments. A location \( p_i \) is chosen by the algorithm. If the angle between segment \( p_{i-1} p_i \) and \( p_i p_{i+1} \) is \( \pi \), the two segments can be combined by removing the central point (see figure 2). No collision checks are necessary in this case.

#### 3.1.2 Redundant points at vertices

A vertex can be deleted if the collision-freeness of the resulting path is still guaranteed. The vertex \( p_i \) can be deleted if the segment \( p_{i-1} p_{i+1} \) is collision-free. This can be accomplished if the smoothing operation described in 3.1.3 is executed with parameter \( \ell_{end} = 1 \). Then only one segment has to be checked.
for each vertex. This method combined with the method from section 3.1.1 represent already a running smoothing algorithm that needs very few collision checks. This can be helpful when dealing with complex scenes where collision checking is computationally expensive.

### 3.1.3 Cutting off sharp vertices

For sharp vertices it is desirable to increase the angle between their two segments. To accomplish this, additional points can be inserted at each of the segments that are adjacent to the sharp vertex. Considering a vertex $p_i$, two points $p_{i,1}$ and $p_{i,2}$ are inserted. The new edge $p_{i,1}p_{i,2}$ must be checked for collision. If a collision occurs, $t$ is increased. Parameter $t$ specifies the number of steps made by this strategy starting with $t = 0$.

$$p_{i,1}(t) = p_i + \frac{p_{i-1} - p_i}{2^t}$$

$$p_{i,2}(t) = p_i + \frac{p_{i+1} - p_i}{2^t}$$

This strategy terminates when $|p_{i,1}p_{i,2}(t)| < \epsilon$ or $t \geq t_{end}$. The variables $\epsilon$ and $t_{end}$ are parameters of the algorithm. In the worst case (i.e., when no solution can be found for the current vertex $p_i$) $\min_i \log_2 \left( \frac{\min_j |p_{i+1} - p_{j-1}p_{j-1} - p_{j+1}|}{\epsilon} \right)$ collision checks for segments have to be calculated.

The methods described in section 3.1.1 and 3.1.2 are special cases of this smoothing operation. For section 3.1.2 the parameter $t_{end} = 1$ and for section 3.1.1 $t_{end} = 1$ and $\angle(p_{i-1}, p_i, p_{i+1}) = \pi$.

### 3.1.4 Choosing a location for improvement

A location best suited for optimization is chosen by a heuristics function. This function should evaluate the highest priority for a location that is linear redundant. Sharp vertices are unfavorable, so they should be assigned a high priority for changing. The heuristics function $h(i) = \frac{|p_{i-1} - p_i| + |p_i - p_{i+1}|}{|p_{i-1} - p_{i+1}|}$ assigns $p_i$ a higher value if the angle between the segments $p_{i-1}p_i$ and $p_ip_{i+1}$ is sharp or equal to $\pi$. If $\angle(p_{i-1}, p_i, p_{i+1}) = \pi$ then $h(i)$ is set to $\infty$. The heuristic value is used to order the candidates. The improvement operation is used at each location that is chosen by $h$. First the special cases described in 3 are checked followed by the phase according to 3.1.3 if $t_{end}$ and $\epsilon$ allow it.

### 3.2 Termination

The termination of the algorithm is trivial if all segments except a straight line from start to goal can be erased, but in each new step a vertex may be added to the path, see 3.1.3. To circumvent an infinite iteration a quality measure allows terminating the algorithm. As a quality measure the sum of angles between subsequent segments is used. If several steps do not lead to an improvement of the whole path, the smoothing can be considered finished.

### 4 TEST CASES AND RESULTS

For tests of the described approach a robot cell has been designed. To demonstrate the algorithms the cell is kept as simple as possible. The only objects are a robot and an obstacle. Three paths have been proposed as the input for the smoothing algorithm: a linear one, a rectangular one and one in shape of an arc. Each of them has been generated using the haptic device. When a specified distance from the previous point is exceeded a new point is created.
a robot along such a path. The time for moving along a path is a good measure for path quality as it is the most crucial factor in production.

![Manual, Haptic, Smooth](image)

Figure 6: Complex test case. Manual: Path was generated using a mouse. Haptic: Path was generated using the Phantom device. Smooth: Path from Haptic that was smoothed.

In table 1 the time the robot needs to move with PTP motions along the result path is displayed. The table shows the time before smoothing (haptic), after smoothing with $t_{end} = 1$ and $t_{end} = \infty$.

The time needed for programming the movement is important. In the complex scene the programming using a mouse in a professional robot simulation application took the authors about 150s. In contrast to that, defining the path with a haptic device took only 15s. The subsequent run of the smoothing algorithm took 75s, which is very long due to the slow distance calculation used as a basis for the collision check on segments in this test. In this example the time benefit of the approach is about 70%. The smoothing can take place in the background, so only 10% is actual user interaction.

<table>
<thead>
<tr>
<th>Test</th>
<th>haptic</th>
<th>$t_{end} = 1$</th>
<th>$t_{end} = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>12.11s</td>
<td>2.35s</td>
<td>2.35s</td>
</tr>
<tr>
<td>Rectangle</td>
<td>19.03s</td>
<td>4.43s</td>
<td>3.73s</td>
</tr>
<tr>
<td>Arc</td>
<td>16.09s</td>
<td>5.07s</td>
<td>4.38s</td>
</tr>
<tr>
<td>Complex</td>
<td>26.48s</td>
<td>5.32s</td>
<td>5.29s</td>
</tr>
</tbody>
</table>

Table 1: Results of tests with smoothing manual haptic generated paths. A simple interpolator is used. Only the acceleration and the maximum speed of the robot are considered.

5 CONCLUSION AND OUTLOOK

The presented paper targets new methods in robot offline programming by integrating haptic input and output. One important factor is path quality. By using e.g. a Phantom device, many points can be created easily that define the desired collision-free path. Though, these paths are not well-suited to be used directly. It is desirable to remove the redundant points. The smoothing algorithm described here increases path quality. At the present state only the geometric path is considered. To evaluate the impact on the process of manual path planning, a survey with users of simulation systems is planned.

REFERENCES


