5-31-2003


Georg F. Mauer
University of Nevada, Las Vegas, georg.mauer@unlv.edu

Follow this and additional works at: https://digitalscholarship.unlv.edu/hrc_trp_fuels

Part of the Controls and Control Theory Commons, Nuclear Engineering Commons, and the Robotics Commons

Repository Citation
Available at: https://digitalscholarship.unlv.edu/hrc_trp_fuels/27

This Report is brought to you for free and open access by the Transmutation Research Program Projects at Digital Scholarship@UNLV. It has been accepted for inclusion in Fuels Campaign (TRP) by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
Design and Evaluation of Processes for Fuel Fabrication

QUARTERLY PROGRESS REPORT #7

UNLV Transmutation Research Program

Prepared by: Georg F. Mauer
Department of Mechanical Engineering
UNLV, Las Vegas, NV 89154-4027
Phone: (702) 895-3830
FAX: (702) 895-3936

Reporting Period: March 1, 2003 through May 31, 2003
Design and Evaluation of Processes for Fuel Fabrication

Summary

The seventh quarter of the project covered the following:

- Mr. Richard Silva continued the development of a simulation model with a Waelischmiller hot cell robot. Rich will continue to develop detailed 3-D process simulation models as his M.Sc. thesis project. Rich is employed with Bechtel at the Yucca Mountain project.
- Mr. Richard Silva presented a paper on hot cell robotics at the ANS student conference in Berkeley, CA.
- Concepts and Methods for Vision-Based Hot Cell Supervision and control, focusing on rule-based object recognition (Ph.D. Student Jae-Kyu Lee)
- Undergraduate student Jamil Renno, developed better control algorithms, and created simulations of pick and place actions for the hot cell manipulator.
- Presentation by Dr. Mauer at U of Nevada, Reno about the Transmuter research at UNLV’s Dept. of Mechanical Engineering on May 2.

Part I Hot Cell Manipulator Simulation

During the present reporting period, the robot simulation model for robot control under Matlab Control software was improved further by student Jamil Renno. Matlab controls the spatial robot model, comprising a geometric model as well as the robot dynamics. Thus a realistic simulation of the forces and torques present during robot motion is being generated. The fuzzy logic controller used during the previous quarter was replaced by a set of six PID controllers. The design parameters of each controller were optimized for minimum overshoot. Care was taken to avoid arm oscillations under any circumstances.

Fig. 2 illustrates the Matlab robot controller, programmed in Simulink format. The block labeled ‘vNPlant’ in Fig. 2 represents the robot dynamics. Figures 3 through 6 show the controller’s some aspects of arm control for joint 1 (robot base): Fig. 3
Figure 2 PID controller for Waelischmiller Robot. Matlab Simulink

Figure 3 Response of Joint 1 (Rotation of Robot Base) to a request to turn 50 degrees
Figure 4  Response of Joint 1 (Rotation of Robot Base) to a request to turn 50 degrees: Initial segment of Fig. 3, showing the gradual acceleration of the arm.

Figure 5  Arm torque of Joint 1 (Rotation of Robot Base) for the motion of Fig. 3
Path Planning:
Figure 6 shows the path planner configuration. Emphasis is placed on smooth trajectories.

Figure 6  Matlab Simulink Program for Path Planning.

Advances in Robot Simulation
In March, a simulation covering six seconds of real time robot motion would take up to three hours. We purchased a Xeon dual processor workstation, and simplified the Matlab numerics.
Through these measures, we were able to increase the simulation speed: the same simulation now takes between 3 and 5 minutes.

**Simulation Work Plan for July through August 2003**

Optimize robot controller for speed, without sacrificing stability or increasing overshoot. Begin the design and testing of a robotic assembly line for pellet insertion: grasp pellets from bin, load specified number of pellets onto tray, and insert the pellet stack into cladding tube, see Fig. 7.

![Figure 7  Concept of Pellet Insertion into Cladding Tube.](image-url)
Part II Object Recognition

As described earlier, an object’s surface is characterized by a series of connected points, see Fig. 2.1 and 2.2. During the reporting period, we worked on knowledge based methods for the retrieval of stored information and for pattern matching. The relationship between a surface and its stored features is represented as a sequence of connected boundary parts, each of them represented as a sequence of junction types in the connectivity table (see Table 2.1)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex</td>
<td>v₁, v₂, … , v₄</td>
<td></td>
</tr>
<tr>
<td>Edge</td>
<td>e₁, e₂, … , e₄</td>
<td></td>
</tr>
<tr>
<td>Junction</td>
<td>v₁: 2-1-4</td>
<td>&lt;&lt; J₁; 2-1-4, null &gt;&gt;</td>
</tr>
<tr>
<td></td>
<td>v₂: 1-2-3, 8-2-1</td>
<td>&lt;&lt; J₂; 1-2-3, 8-2-1 &gt;&gt;</td>
</tr>
<tr>
<td></td>
<td>v₃: 4-3-2, 4-3-6</td>
<td>&lt;&lt; J₃; 4-3-2, 4-3-6 &gt;&gt;</td>
</tr>
<tr>
<td></td>
<td>v₄: 3-4-1, 3-4-5</td>
<td>&lt;&lt; J₄; 3-4-1, 3-4-5 &gt;&gt;</td>
</tr>
<tr>
<td>Loop</td>
<td>2-1-4</td>
<td>&lt;&lt; L₁; 2-1-4 &gt;&gt;</td>
</tr>
<tr>
<td></td>
<td>1-2-3</td>
<td>&lt;&lt; L₂; 1-2-3 &gt;&gt;</td>
</tr>
<tr>
<td></td>
<td>4-3-2</td>
<td>&lt;&lt; L₃; 4-3-2 &gt;&gt;</td>
</tr>
<tr>
<td></td>
<td>3-4-1</td>
<td>&lt;&lt; L₄; 3-4-1 &gt;&gt;</td>
</tr>
<tr>
<td>Surface</td>
<td>Sₐ</td>
<td>&lt;&lt; Sₐ; L₁, L₂, L₃, L₄ &gt;&gt;</td>
</tr>
<tr>
<td>Feature</td>
<td>Sₐ, S₉, S₇</td>
<td>&lt;&lt; F; Sₐ, S₉, S₇ &gt;&gt;</td>
</tr>
</tbody>
</table>
For example, in Figure 2.1, a relative relationship in the surface $S_B$ to surface $S_A$ is $<< S_B$-to-$S_A$; $J_3, J_4 >>$ or, in case of surface $S_A$ only, $<< S_A; L_1, L_2, L_3, L_4 >>$. An knowledge base type representation of loop and junction for the feature of Figure 2.1 is shown in Figure 2.3 at left.

A set of initial rules for pattern matching has been developed and verified.

**Recognition Process using Rule-Based System**

### 2.2 Rules and Decision Functions for Pattern Recognition

Decision functions have been also defined as computer code for implementation and testing.

**Rule 1**: edge match (low-level rule)

Let $VIEW_M$ as a view type of model image $M$ and let $T\_VECTOR$ be the set of test vector in any given test scene $T$, and let $EDGE\_VIEW\_M$ be the set of magnitude of edge from $VIEW_M$. Then an instance of $EDGE\_VIEW\_M$ to $T\_VECTOR$, INST1, becomes

$$INST1: T\_VECTOR \rightarrow EDGE\_VIEW\_M$$

The decision function of rule 1 is comparison of edge magnitude between model image $VIEW_M$ and test scene $T$ as follows:

**Decision function** of rule 1

$$\text{if } \left( \frac{|EDGE\_VIEW\_M|}{T\_VECTOR} \leq tol \right)$$

$$\text{then} (T\_VECTOR \rightarrow T\_CO\_EDGE)$$

There is partial interpretation to classify loops from co-edges in the test scene after rule 1 has been executed and collected their correct matches. The co-edges could be more than three edges during the implementation of rule one, and in that case, they must be split away as three non-collinear point sets, loop.
Partial interpretation (I)

Function: classifying loops from co-edges

Initiation timing: after rule 1 has been executed

\[ P_{\text{INT1}}: T_{\text{CO\_EDGE}} \rightarrow T_{\text{LOOP}} \]

Rule 2: loop matching (intermediate level rule)

Let loops from a view type of model image VIEW\_M as LOOP\_VIEW\_M and let T_LOOP be the set of loops in T. Then an instance of LOOP\_VIEW\_M to T_LOOP, INST2, becomes

The decision function of rule 2 is comparison of inner angle between LOOP\_VIEW\_M and T_LOOP after partial interpretation 1 has accomplished and loop from test scene has been classified.

Decision function of rule 2

\[
\text{if} \left( \frac{\angle LOOP\_VIEW\_M}{\angle T\_LOOP} \leq \text{tol} \right)
\]

\[
\text{then} \left( T\_LOOP \rightarrow LOOP\_VIEW\_M \right)
\]

Rule 3: surface invariance (advanced level rule)

Let all the transformation information about surface from a view type of model image VIEW\_M as SURFACE\_VIEW\_M and let T_SURFACE be the set of all the transformation information about surface of T. Then an instance of SURFACE\_VIEW\_M to T_SURFACE, INST3, becomes

\[ \text{INST3}: T\_SURFACE \rightarrow SURFACE\_VIEW\_M \]

The decision function of rule 3 is comparison of transform invariance of surface between model image and test scene.

Decision function of rule 3

\[
\text{if} \left( U^A G^A P = U^B G^B P \right)
\]

\[
\text{then} \left( T\_SURFACE \rightarrow SURFACE\_VIEW\_M \right)
\]

Here the transform \( G \) and point \( P \) on loop pairings vector in the tests scene are the same as shown in Figure 2.2.
Partial interpretation (II)

Function: determine occlusion by degree of uncertainty (DOU)

\[ \text{DOU} = \sum_{i=0}^{m} m_i - \sum_{j=0}^{n} t_j \]

where \( t \): number of loops from the test scene
\( m \): number of loop in the model image

Initiation timing: after rule 3 has been executed

Rule 4: feature neighborhood (advanced level rule II)

Let any surface from a view type of model image \( \text{VIEW}_M \) as \( \text{FEATURE}_\text{VIEW}_M \) and let \( T_\text{FEATURE} \) be the set of surfaces in the test scene \( T \) as newly found. Then an instance of \( \text{FEATURE}_\text{VIEW}_M \) to \( T_\text{FEATURE} \), \( \text{INST}_4 \), becomes \( \text{INST}_4 : T_\text{FEATURE} \rightarrow \text{FEATURE}_\text{VIEW}_M \)

The decision function of rule 4 is feature neighborhood match using junction connectivity.

Decision function of rule 4

\[ \text{if} (\text{JUNCTION}_\text{VIEW}_M \supseteq T_\text{JUNCTION}) \text{then} \]
\[ (T_\text{FEATURE} \rightarrow \text{FEATURE}_\text{VIEW}_M) \]

To recognize 3D features from the test scene, there are two partial interpretations after we accomplish rule 4. One is super perimeter and the other one is backward chaining. These two interpretations will not always available but when it comes to knowledge-based system, the iteration becomes much faster.

Partial interpretation (III)

Function: super perimeter search

Initiation timing: whenever super perimeter found

Partial interpretation (IV)

Function: parallel process and backward chaining

Initiation timing: whenever surface is found

Verification of Recognition Rules

The proper function of the rules was tested on synthetic objects (see Fig. 2.4) and on photographed overlapping objects (grey-tone images, see Fig. 2.5). The two objects in Fig. 2.5 were photographed using a CCD camera in the robotics lab.
Figure 2.4 2D recognition
Synthetic Image

Figure 2.5 3D recognition
Grey tone CCD image of Physical Objects
Appendix  Selected Simulation Results

The simulations presented below were developed on a fast dual Xeon processor workstation using ProEngineer solid modeling software in conjunction with MSC VisualNastran4D and Matlab Simulink for control. A single simulation takes approx. 5 minutes. On the previous machine, equipped with a Pentium IV processor at 1.7 GHz, a typical single run would take 6 to 10 hours. Together, the software tools create realistic 3D animations, as well as accurate calculations of the dynamics of all components, loads, collisions, and other aspects of the hot cell operations. The three figure sequence below illustrates a Waelischmiller robot picking a part from the green table, and placing it onto the grey table in the background. The robot’s 6 axes are controlled using a PID algorithm (one controller for each joint) in Matlab Simulink.

Figure A1  Interactive GUI process simulation: Waelischmiller robot picks part from green table. Created by Jamil Renno with Visual Nastran. 6-axes robot PID control in Matlab Simulink
**Figure A2** Interactive GUI process simulation: Part being moved to grey table. Created by Jamil Renno with Visual Nastran. 6-axes robot PID.

**Figure A3** Interactive GUI process simulation: The part has arrived at its destination: the grey table. Created by Jamil Renno with Visual Nastran.
References