Design and Evaluation of Processes for Fuel Fabrication

QUARTERLY PROGRESS REPORT #6

UNLV AAA University Participation 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: December 1, 2002 through February 28, 2003

Design and Evaluation of Processes for Fuel Fabrication

Summary

The sixth 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.
- Concepts and Methods for Vision-Based Hot Cell Supervision and control (Ph.D. Student Jae-Kyu Lee)
- An undergraduate senior student in mechanical engineering, Mr. Jamil Renno, was hired to develop the simulation model for the hot cell manipulator.

Part I   Hot Cell Manipulator Simulation

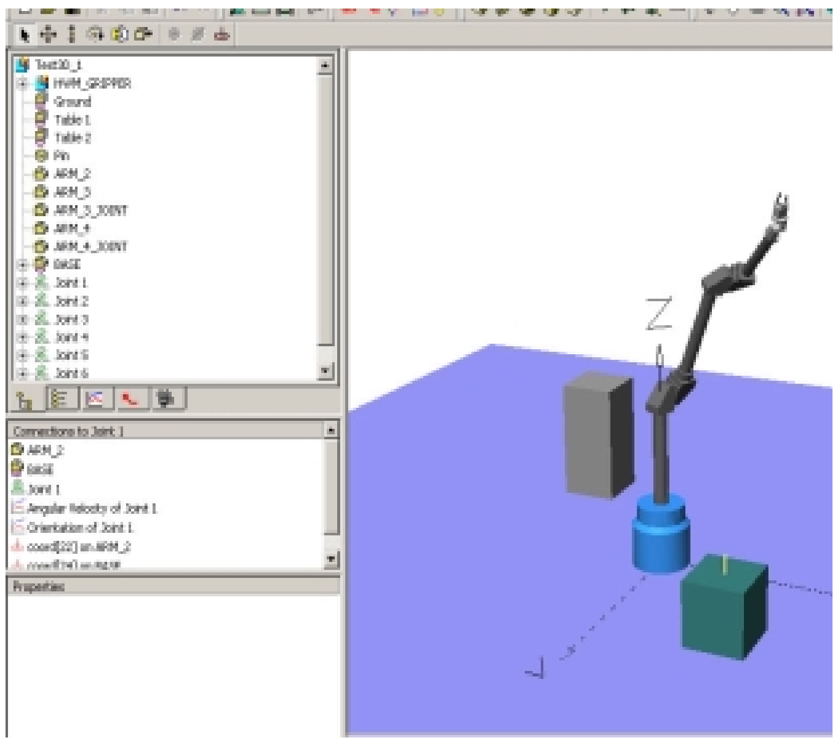
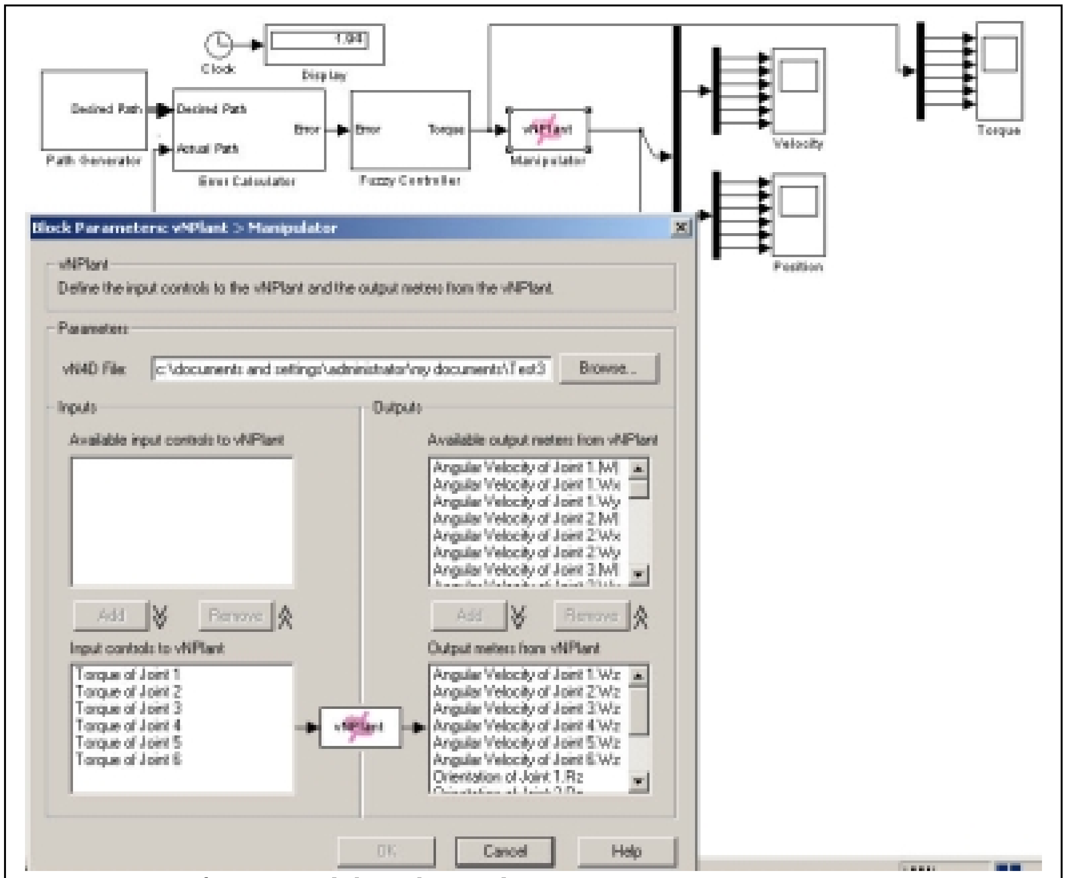


Figure 1 Hot Cell Robot Simulation (Waelischmiller)

As previously reported, graduate student Richard Silva developed a 3D simulation model for a Waelischmiller-type hot cell robot (see Fig. 1). During the present reporting period, the robot simulation model for robot control was coupled with Matlab Control software by Mr. Jamil Renno. Matlab is now interfacing with, and controlling the spatial robot model. Said robot model comprises a geometric model as well as the modeling of the robot dynamics. Thus a realistic simulation of the forces and torques present

during robot motion is being generated.
Fig. 2 illustrates the interaction between Visual Nastran4D and Matlab (see Fig. 2). The block labeled ‘vNPlant’ in the top center of Fig. 2 represents the robot dynamics. The textbox labeled ‘Block parameters…’ lists variables in Visual Nastran4D that are accessed by Matlab.



Block Parameters: vNPlant > Manipulator

vNPlant
Define the input controls to the vNPlant and the output meters from the vNPlant.

Parameters
vN4D File: c:\documents and settings\administrator\my documents\Test3

Inputs
Available input controls to vNPlant

Input controls to vNPlant
Torque of Joint 1
Torque of Joint 2
Torque of Joint 3
Torque of Joint 4
Torque of Joint 5
Torque of Joint 6

Outputs
Available output meters from vNPlant
Angular Velocity of Joint 1.|W|
Angular Velocity of Joint 1.Wx
Angular Velocity of Joint 1.Wy
Angular Velocity of Joint 2.|W|
Angular Velocity of Joint 2.Wx
Angular Velocity of Joint 2.Wy
Angular Velocity of Joint 3.|W|

Output meters from vNPlant
Angular Velocity of Joint 1.Wz
Angular Velocity of Joint 2.Wz
Angular Velocity of Joint 3.Wz
Angular Velocity of Joint 4.Wz
Angular Velocity of Joint 5.Wz
Angular Velocity of Joint 6.Wz
Orientation of Joint 1.Rz

Figure 2 Interfacing Matlab with Visual Nastran4D.

Controller Design

The Waelischmiller manipulator has six rotary joints. Each joint is controlled by a distinct fuzzy-logic controller, for a total of six fuzzy-logic controllers. The input for each fuzzy controller is its respective angle error, i.e. its deviation from the desired reference value. We also control the deviation of the angular velocity from the reference for each joint. The controller output is the torque applied at each joint.

Path Planning:
Most methods use smooth polynomials. A widely used approach in robotics is to follow the 'bang-bang' motion profile. Our bang-bang motion profile has these components: accelerate smoothly, drive at steady speed, and decelerate smoothly. Fig. 3 shows an example of 'band-bang' robot joint motion.

Actual Path of motion:
The actual path of motion is provided through the *VNPlant* block in *Simulink*. This block provides the actual angle and the actual angular velocity of each joint.
The error is defined as: Error = Desired Path – Actual Path

Structure of Fuzzy Control:
Each fuzzy controller employs five Gaussian membership functions. The membership functions are defined as follows:

NB: Negative Big
NS: Negative Small
Z: Zero
PS: Positive Small
PB: Positive Big

Those linguistic values are scaled numerically as needed so that each controller does perform the requested task well. The dynamics of a properly designed fuzzy logic controller are roughly equivalent to those of a classical PID controller.
Table 1 shows the matrix of rules used to develop the controllers:

Table 1 Rule Matrix of Fuzzy Logic Controller

Velocity Error	Displacement Error				
	NB	NS	Z	PS	PB
NB	NB	NB	NS	NS	Z
NS	NB	NS	NS	Z	PS
Z	NS	NS	Z	PS	PS
PS	NS	Z	PS	PS	PB
PB	Z	PS	PS	PB	PB

The row and column entries denote the inputs, while the inner elements denote the controller output. For example: if the velocity error is NS (negative small) and the displacement error is PB (positive big), then the torque is PS (positive small).

Figure 4 shows the fuzzy logic controller configuration, while Fig. 5 illustrates the generation of the bang-bang velocity profile.

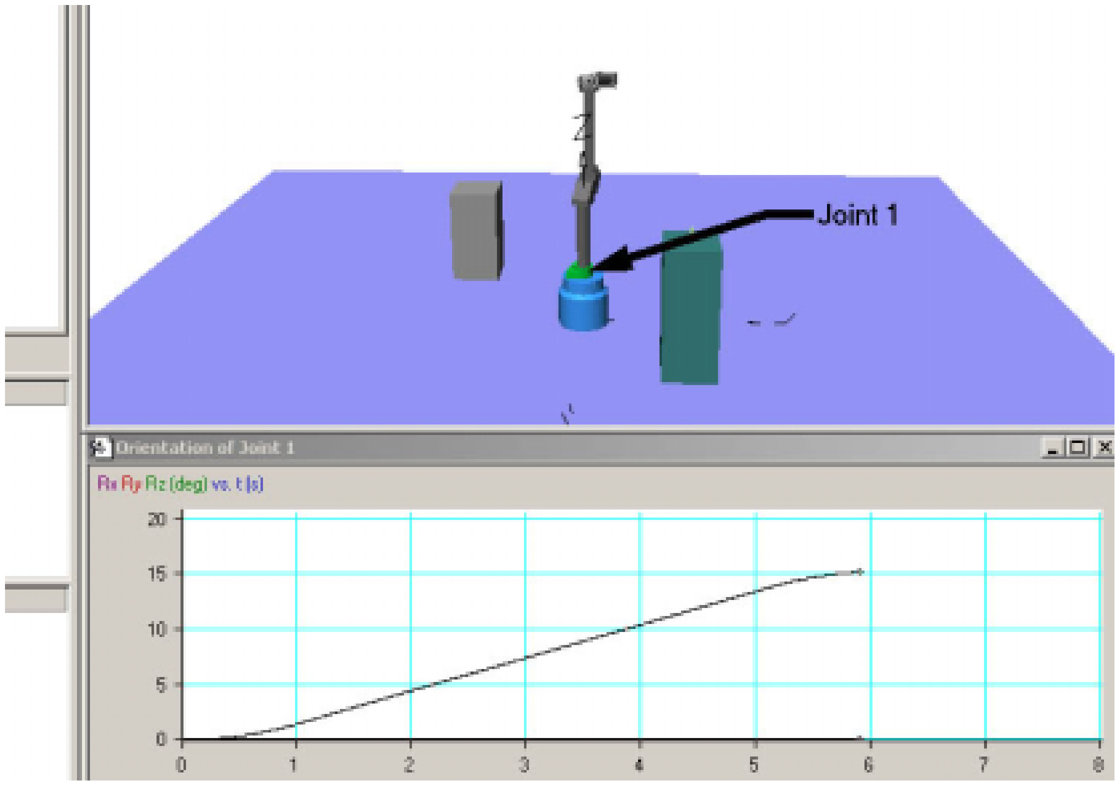


Figure 3 ‘Bang-Bang’ motion of Joint 1 (Rotation of Robot Base)

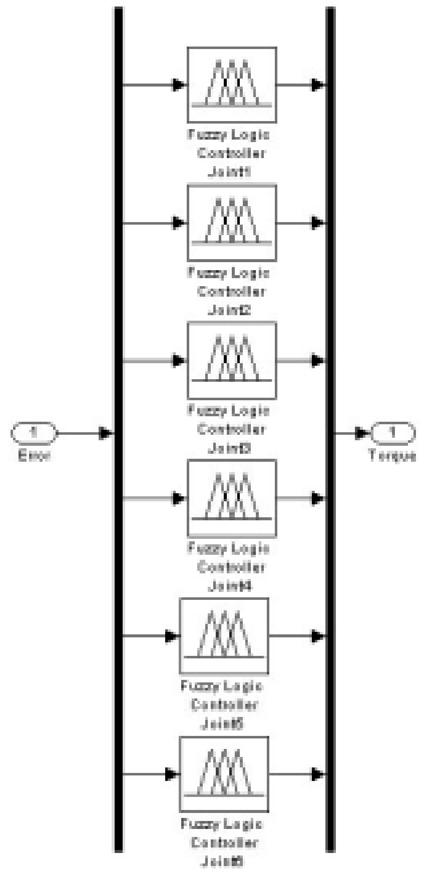

1
Error
Fuzzy Logic Controller Joint1
Fuzzy Logic Controller Joint2
Fuzzy Logic Controller Joint3
Fuzzy Logic Controller Joint4
Fuzzy Logic Controller Joint5
Fuzzy Logic Controller Joint6
1
Torque

Figure 4 Matlab Simulink Model of Fuzzy Logic Robot Control

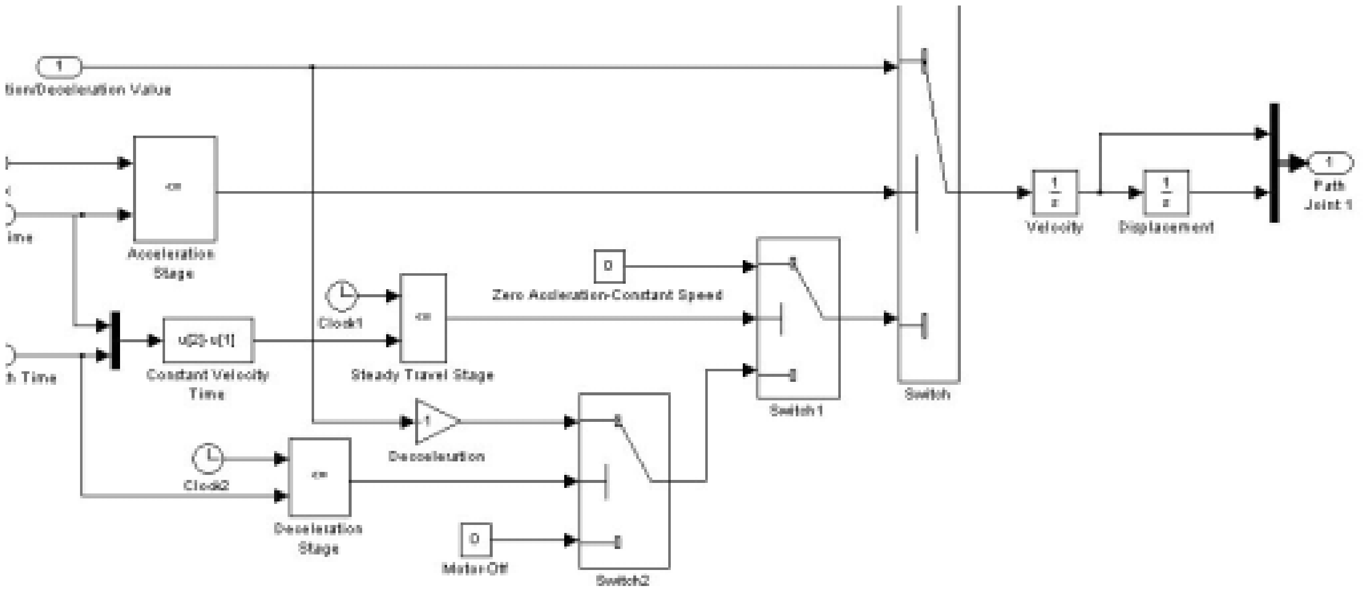


Figure 5 Matlab Simulink Program for Bang-bang Velocity Reference Generation.

Problems with the Simulation

While the simulation process generally works flawlessly, the simulation time has risen considerably as more details were added to the simulation. A simulation covering six seconds of real time robot motion can take up to three hours. We are investigating options to increase the speed, especially since we expect to add significantly more complexity to the simulation as the project progresses.

Part II Object Recognition

Concept – Objects are recognized from CCD camera images based on their geometrical features. Machine recognition is the matching of an image pattern with known patterns generated by the same object class and stored in a database. Since detected patterns vary considerably with distance, viewing angle, lighting conditions, and also occlusions, a systematic method for reducing and organizing the vast quantities of possible reference data would be helpful. Our approach identifies objects by their contours. The database stores only sets of characteristic points, such as edges and corners. Indexing employs a priori stored information about the object models, in order to quickly eliminate non-compatible model-scene feature matches during recognition. Hence, only the most feasible matches are considered, that is, the matches where the model features could have projected to the scene features. Indexing-based methods usually employ a hash scheme to efficiently store and retrieve information about the models into a *hash table*. During preprocessing, groups of model features are considered, and a description for each is computed and stored in the indexed location. During recognition, groups of scene points are used to access the hash table. An unknown object is identified using geometric interpolation between a finite set of stored views. A more detailed discussion is contained in the appended CAINE 2002 conference paper. A nearest neighbor (NN) algorithm establishes the best match between candidate object and the stored views in the database. The recognition of objects in full (unoccluded) view requires as a minimum a

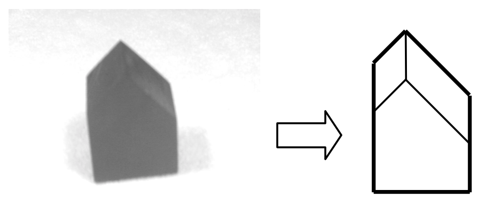


Fig. 5 Sub feature clustering of a 3D object in Fig. 1. the object is characterized by four features including pentagon (bold faced) that represents the whole shape of object.

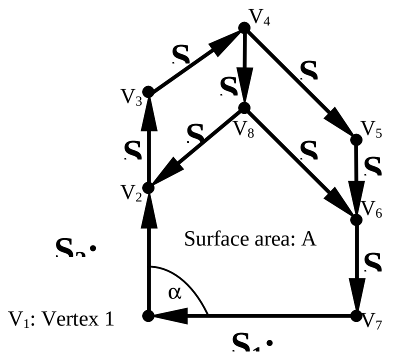
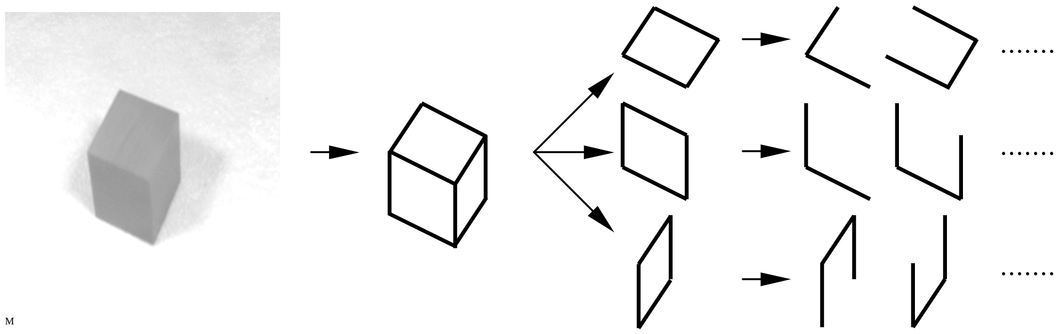


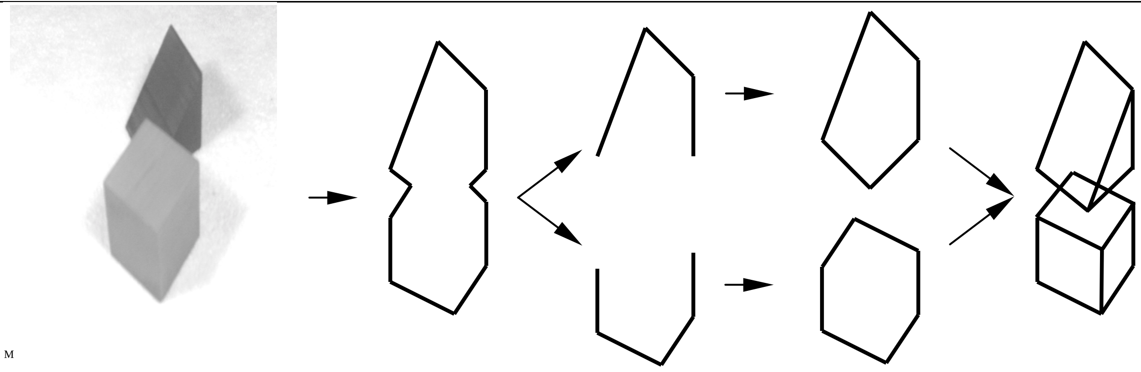
Fig. 6 Feature vector indexing for four sub features of object in Fig. 5.

match of three characteristic points with their reference in the database.



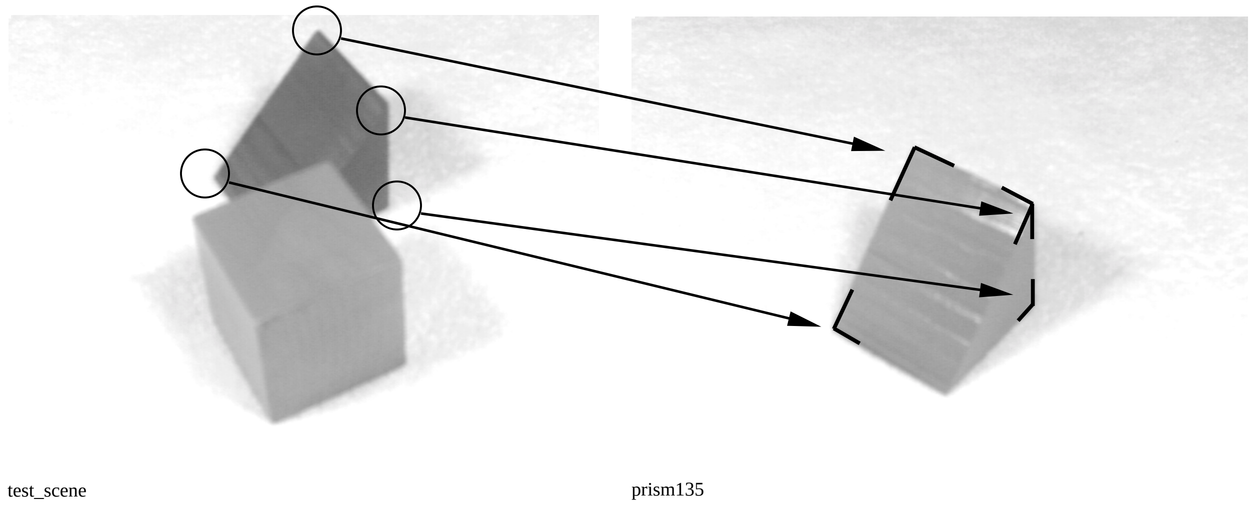
(a) Model image	(b) Model image shape feature	(c) Model sub-features	(d) Clustered vectors of sub-features

Fig. 7 Illustration of indexing and clustering of model feature and sub-features. Here we have for cubic object. Every model image will be classified as the same way and stored in the database.

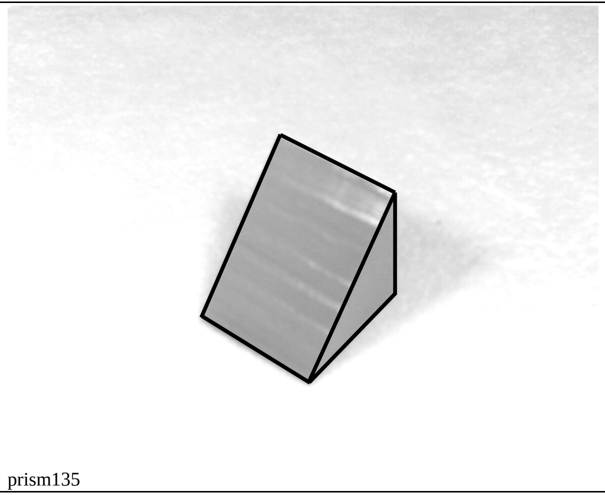


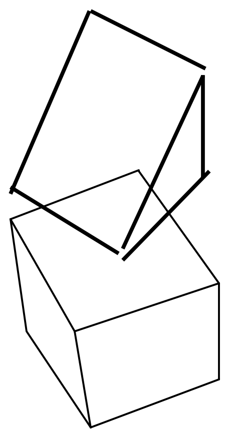
(a) Test scene	(b) Test scene shape feature	(c) Developing hypothesis by model sub feature clusters	(d) Recovered model shape features	(e) Regenerate test scene feature

Fig. 8 Schematic procedure of recovering hidden prism behind cubic in occluded case using feature-based indexing.



(a) Original test scene	(b) Point and junction match





(c) Nearest feature selected from model image database	(d) Regenerated test scene
Fig. 9 Recovery of occluded object from test scene image which taken from arbitrary viewing angle The lines in (d) are drawn manually for visualization only	

Feature Vector Indexing

The example of Fig. 5 illustrates how a 3D object may be segmented into a set of simple geometric features: here two rectangular and one small pentagon and one large pentagon (in bold) that represent the whole body shape. Each feature is stored with its vector segmentation

and geometry information (magnitude, inner angle, etc). Fig. 6 shows the resulting cluster feature-based indexing vectors that are entered into the object database. Figures 7 and 8 present examples of model generation and recovery of hidden features, respectively. Fig. 9 illustrates the identification of partially occluded objects.
We are presently in the process of creating and testing further software for automatic feature recognition and recovery.

References

1. Chin, Ronald T.; Dyer, Charles R.; “Model–Based Recognition in Robot Vision” Computing Surveys, Vol. 18, No. 1, pp. 67-103, March 1986.
2. Grimson, W. E. L.; Huttenlocher, D. P.; “On the Sensitivity of Geometric Hashing” Proceeding on International Conference of Computer Vision, pp. 334-338, 1990.
3. Lamdan, Yehezkel; Schwartz, Jacob T.; Wolfson, Haim J.; “Affine Invariant Model-Based Object Recognition” IEEE Transactions on Robotics and Automation, Vol. 6, No. 5, pp. 578-589, October 1990.
4. Stein, F.; Medioni, G.; “Structural Indexing: Efficient 2D Object Recognition” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 12, pp. 1198-1204, December 1992.
5. Stein, F.; Medioni, G.; “Structural Indexing: Efficient 3D Object Recognition” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, pp. 125-145, February 1992.
6. Califano, Andrea; Mohan, Rakesh; ”Multidimensional Indexing for Recognizing Visual Shapes” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 4, pp. 373-392, April 1994.
7. Beis, Jeffrey S.; Lowe, David G.; “Indexing without Invariant in 3D Object Recognition” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 10, pp. 1000-1015, October 1999.
8. Ben-Arie, Jezekiel; ”The probabilistic Peaking Effect of Viewed Angles and Distances with Application to 3-D Object Recognition” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 8, pp. 760-774, 1990.
9. Burns, J. Brian; Weiss, Richard S.; Riseman, Edward M.; ”View Variation of Point-Set and Line-Segment Features” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 1, pp. 51-68, January 1993.
10. Olson, Clark F.; “Probabilistic Indexing for Object Recognition” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 5, pp. 518-522, May 1995.

11. Ullman, Shimon; Basri, Ronen; “Recognition by Linear Combinations of Models” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 10, pp. 992-1006, October 1991.
12. Bebis, George; Georgiopoulos, Michael; Shah, Mubarak; Vitoria Lobo, Niels; “Indexing Based on Algebraic Functions of Views” Computer Vision and Image Understanding, Vol. 72, No. 3, December, pp. 360-378, 1998.
13. Tveter, R. Donald; “The Pattern Recognition Basis of Artificial Intelligence”, IEEE Computer Science Society, 1998.
14. Medioni, Gerard, Yasumoto, Yoshio; “Corner Detection and Curve Representation Using Cubic B-Spines”, Computer Vision, Graphics, And Image Processing 39, 267-278(1987).