An Interactive Framework for Acquiring Vision Models of 3-D Objects From 2-D Images

Yuichi Motai, *Member, IEEE,* and Avinash Kak

***Abstract*—This paper presents a human-computer interaction (HCI) framework for building vision models of three-dimensional (3-D) objects from their two-dimensional (2-D) images. Our framework is based on two guiding principles of HCI: 1) provide the human with as much visual assistance as possible to help the human make a correct input; and 2) verify each input provided by the human for its consistency with the inputs previously provided. For example, when stereo correspondence information is elicited from a human, his/her job is facilitated by superimposing epipolar lines on the images. Although that reduces the possibility of error in the human marked correspondences, such errors are not entirely eliminated because there can be multiple candidate points close together for complex objects. For another example, when pose-to-pose correspondence is sought from a human, his/her job is made easier by allowing the human to rotate the partial model constructed in the previous pose in relation to the partial model for the current pose. While this facility reduces the incidence of human-supplied pose-to-pose correspondence errors, such errors cannot be eliminated entirely because of confusion created when multiple candidate features exist close together. Each input provided by the human is therefore checked against the previous inputs by invoking situation-specific constraints. Different types of constraints (and different human-computer interaction protocols) are needed for the extraction of polygonal features and for the extraction of curved features. We will show results on both polygonal objects and object containing curved features.**

***Index Terms*—Human-computer interaction, pose–to–pose correspondence, 2-D, 3-D, vision models.**

I. Introduction

IT IS NOW generally believed that model based vision of the kind reported in [1], [3], [5] cannot always be driven by CAD models of objects. Mechanical CAD models do not always provide a good representation of objects for recognition by a computer vision system. Since a CAD model must by definition be a complete geometric representation of an object, the resulting representations can be excessively complex and geometrically too rich for computer vision work.

On the other hand, the representation for a vision system needs to be sufficient only for the purpose of recognition and pose estimation. That is, the representation used should provide just sufficient discriminatory power to differentiate between a given object and all the other objects that a vision system is expected to see. Obviously, if a vision system is expected to see only one type of a rigid object in a clutter-free environment, the needed representation can be very simple and consist of just a small number of features that would be needed for pose calculation. We will refer to the representation needed for recognition and pose calculation as a vision model—a term that has been used before by many other researchers.

What that means is that, unlike a CAD model which must be geometrically complete and precise, what vision model one uses depends on a host of factors such as what other objects the vision system will be seeing, the visual similarity between the objects, the nature of the clutter in the background, etc.

It therefore stands to reason that while it may be possible to derive a vision model from a CAD model, a superior strategy might consist of constructing a vision model directly from the object itself. Ideally, one would want to show N objects to a vision system, each in multiple poses, tell the vision system that those N objects are to be treated as visually different, and then have the vision system figure out on its own how to differentiate between the objects and between different poses for a given object.

But, as is so well known in the vision community, we are far, far from achieving this ideal. In the meantime, all we can hope for is that it would be possible for a computer to construct vision models with human assistance—hopefully minimal human assistance. For the foreseeable future, one would want to be able to develop easy-to-use graphical interfaces and user-interaction protocols that would permit a human to help the computer with solving those aspects of model-building that cannot yet be fully automated.

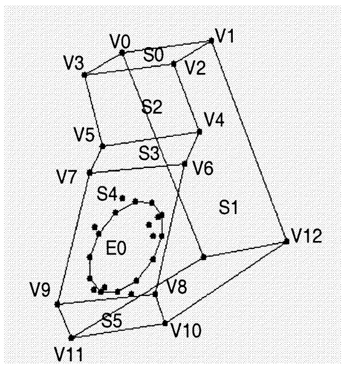
For sure, the development of such graphical user interfaces and human-computer interaction protocols has received much attention from industry [2], [12], [21] and academia [4] in recent years. These interfaces, intended primarily for animators, architects, forensic specialists, and so on, allow one to construct a three-dimensional (3-D) model from multiple two–dimensional (2–D) images taken from different viewpoints. There is an important reason for why these prior systems cannot be used in robotic and industrial computer vision applications: *The end-goal of these systems is the graphical rendering of a reconstructed 3-D object. This means that the notion of a feature that might play a critical role in object-recognition for bin-picking or in vision-guided robotic assembly is not central to the overall process of 3-D object reconstruction.*

Vis-a-vis the systems described in [2], [4], [12], [21], the work described in this paper is geared primarily toward robotic vision applications where the reconstructed 3-D models must be rich in features. Our human-computer interaction protocol

Manuscript received December 12, 2002; revised June 4, 2003. This work was supported by the Advanced Manufacturing Technology Development, Ford Motor Company. This paper was recommended by Associate Editor Q. Zhu.

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: motai@purdue.edu, kak@purdue.edu).

Digital Object Identifier 10.1109/TSMCB.2003.817085



(a)

Feature Surfaces	Vertex Entities	Area (mm^2)	Perimeter (mm)	Shape Complexity
Polygon S0	V0 V1 V2 V3	3045	244	4.42
Polygon S1	V1 V2 V4 V6 V8 V10 V12	8769	441	4.71
Polygon S2	V3 V2 V4 V5	3828	262	4.23
Polygon S3	V4 V6 V7 V5	1586	212	5.32
Polygon S4	V6 V8 V9 V7	5005	542	7.66
Polygon S5	V8 V10 V11 V9	1144	202	5.97
Ellipse E0	8 cardinal points	2827	188	3.54
Vertices	Adjacent Vertices	X Coordinates	Y Coord.	Z Coord.
V0	V1 V3 V13	34.9	46.6	124.9
V1	V0 V2 V12	-46.4	35.3	125.1
V2	V1 V3 V4	-44.8	1.7	125.5
V3	V0 V2 V5	39.3	10.9	125.1
V4	V2 V5 V6	-42.7	-3.4	83.5
V5	V3 V4 V7	47.7	8.4	84.2
V6	V4 V7 V8	-42.1	-19.5	74.0
V7	V5 V6 V9	45.8	-8.8	71.6
V8	V6 V9 V10	-31.0	-81.0	18.2
V9	V7 V8 V11	56.0	-67.5	15.1
V10	V8 V11 V12	-24.9	-83.1	-1.6
V11	V9 V10 V13	52.6	-64.6	-3.6
V12	V1 V10 V13	-30.9	37.0	-1.9
V13	V1 V11 V12	43.7	57.1	-2.7

(b)

Fig. 1. Gometrical representation of a typical 3-D polyhedral object. (a) A wireframe represented form of the reconstructed 3-D vision model with labeled features (b) Correspondent features and their attributes.

makes it convenient for a human to assist the computer with the grouping processes required for the formation of these features. More specifically, our system requires a human to place a 3-D object under a multicamera system in its various poses and to then

- help the computer with the extraction of the visually significant planar and curved features;
- help the computer establish stereo correspondences if needed;
- help the computer establish pose-to-pose correspondences when needed;

so that the computer can build a 3-D vision model of the object. As we will describe in greater detail in the next section, the vision system records five different images for each pose of the object chosen by the human. These five images consist of a central main view and four ancillary views. This data configuration, which differs markedly from what is employed in the systems described in [2], [4], [12], [21], is necessitated by the requirements of the domain—accurate reconstruction of the features of industrial objects in a 3-D vision model. The cited systems try to reconstruct a 3-D graphical model by a global integration of all the views all at the same time. On the other hand, our human-computer interaction protocol seeks to localize the feature with high-accuracy in each view that is considered primary for that feature by the human—the localization achieved with the help of the other four views.

To quickly show a result before launching into a detailed discussion of our system, Fig. 1(a) shows a reconstructed vision model. For the sake of visual display, we only show a wire-frame representation of the reconstructed model. But note that all of visually significant features in the reconstructed model are attributed. The attributes associated with the various features in the reconstructed model are shown in the form of a table in Fig. 1(b). Whereas the attributes *area* and *perimeter* have their usual meanings, the attribute *shape complexity* was calculated as $perimeter/\sqrt{area}$.

Learning 3-D models from 2-D images is much more challenging than doing the same from range images. A high-quality range scanner will faithfully capture most if not all of the external surfaces of a wide variety of 3-D objects. Constructing a 3-D model from the range maps taken from different viewpoints then consists of establishing correspondence between the same surfaces in the different range maps and "stitching" the surfaces together into a full 3-D model. This is what has been done in many recent notable contributions that have brought computer vision and computer graphics together for the construction of 3-D models of the statue of Michelangelo's David at the museum [16] and of the statue of Great Buddha at Kamakura temple [19]. If this learning process is also supposed to teach the computer how best to distinguish between the constructed models for recognition and pose calculation, the model building approach can be combined with the decision-tree based learning approach as advanced in the MULTI-HASH system [9].

Ours is by no means the first contribution in constructing 3-D models from 2-D images. For very simple 3-D shapes, such as cylinders, there is the contribution of Ogawara *et al.* [20] in which prior knowledge of the dimensions and a color histogram based segmentation of the round surface of a cylinder is used to construct a vision model of the cylinder for its future recognition using color cues. An extension of this idea to simple polyhedral shapes, again assuming that the basic dimensions of the model

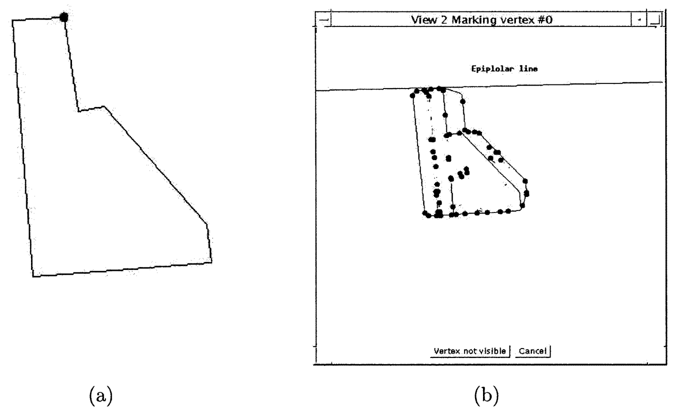


Fig. 2. Stereo correspondence for a human with an epipolar line.

are already known, is presented by Kuniyoshi *et al.* [15]. Another related contribution is from Heuel and Nevatia [10] in the context of mobile robotics vision in which 2-D images are used for the construction of a 3-D model of a building whose basic geometry and layout are already known. The reader is also referred to a human-assisted model construction system by Fracois and Medioni [7] that builds a 3-D model from a single image by applying NURBS fitting to human-supplied control points.

A human-computer interaction framework for constructing vision models of 3-D objects from 2-D images must address the problems caused by the fact that low-level feature extraction algorithms are notoriously ill-behaved. A classic example of this is the output of an edge detector. In order not to miss important edges, one usually tries to use detection thresholds that are liberal, leading to cluttered edge maps in most situations. So when human assistance is sought in identifying features in such maps or in establishing stereo correspondences between two such edge maps, we have to accept the high possibility that the human might make an error. We have tried to alleviate such problems as follows.

- Providing the human with as much visual assistance as possible for interaction. As a case in point, when the system asks the human for identifying a stereo correspondence, the system superimposes an epipolar lines on the image in which the correspondence is sought. What's interesting is that such facilities do not eliminate errors—they only reduce the probability of error. For example, shown in Fig. 2(a) is an image with a highlighted feature point. The computer would like a stereo correspondence for this feature point in a different viewpoint image that is shown in Fig. 2(b). To help the human, the computer superimposes on the image in (b) the epipolar line for the highlighted feature point in (a). As the reader can see, despite the epipolar line, there are multiple pixel candidates in (b) for the highlighted feature point in (a). The confusion created by the presence of such multiple candidates justifies our second step below.
- Each input by the human is verified for its consistency with the previously supplied inputs.

We will identify these two aspects of our system in each of the interaction phases presented in the rest of this paper.

In the rest of this paper, in Section II we first give the reader an overview of the human-computer interaction in our system. Subsequently, we describe in greater detail the various functional modules of the system. Along those lines, Section III discusses how polyhedral features are extracted in 2-D and reconstructed in 3-D; this section also shows how multiple poses are merged for the case of polyhedral features. Section IV then does the same for general elliptical features. Subsequently, Section V presents the results of an evaluation study where we report on the accuracy with which polyhedral and curved-feature models are reconstructed by our system. Finally, Section VI concludes our paper by summarizing what we have learned and what remains to be done.

II. Overview of the Interaction Framework

Fig. 3 shows separately the responsibilities of the computer and those of the human. The responsibilities assigned to each agent exploit the unique strengths of that agent. For the most part, the computer carries out actions that require extensive numeric or symbolic computations. On the other hand, the human carries out actions that require perceptual abilities that a machine cannot yet be endowed with.

The model building process begins with the human placing the object under the camera system in each of its stable poses. The computer manipulates the robotic arm holding the camera and collects five different images of the object from a central view and from four other peripheral views. In the rest of this paper, we will refer to the image taken from the central overhead view as the View_0 image. The other four images will be referred to as View_1, View_2, View_3, and View_4 images (see Fig. 4).

The computer then applies an edge detector and vertex detector to the images. Next, the human helps the computer by accepting some or all of the vertex features identified by the computer and, if necessary, by identifying additional vertex features. Subsequently, the human helps the computer with the organization of those feature points into larger level features that can be polygonal and curved. To create a 3-D representation of those features, the computer then seeks human help for establishing stereo correspondences for feature points. To make the job of the human easier, the computer draws epipolar lines on the images. The 3-D representations of the object features created in this manner are subject to various verification tests. After all the information that can be gleaned from the five images in one stable pose is acquired, the human repeats the process for the next stable pose. Later, the human also helps the computer with establishing pose-to-pose correspondences so that the computer can integrate all of the 3-D features in all of the stable poses into a single model.

Fig. 5 shows the various buttons and the clickable icons of the Graphical User Interface that is used to orchestrate the interaction between a human and the computer for the purpose of object modeling. The second row of buttons allows a human to execute the following steps after the object is placed in the work area of the vision system.

- Acquire a sequence of images, usually five, from five different viewpoints.

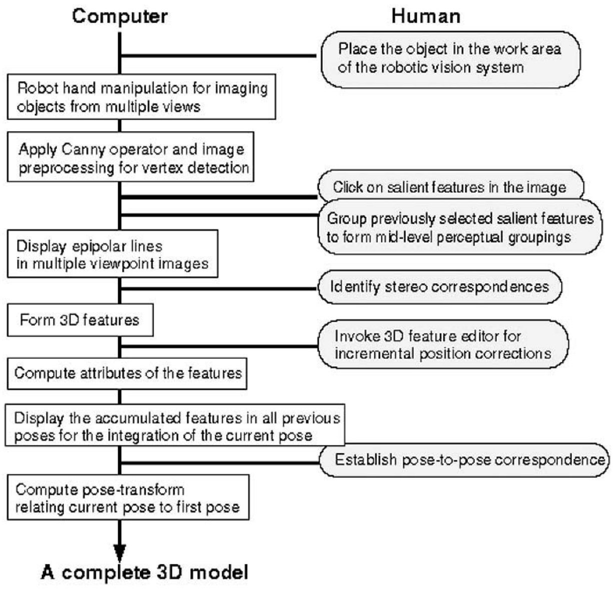

Computer
Human
Place the object in the work area of the robotic vision system
Robot hand manipulation for imaging objects from multiple views
Apply Canny operator and image preprocessing for vertex detection
Click on salient features in the image
Group previously selected salient features to form mid-level perceptual groupings
Display epipolar lines in multiple viewpoint images
Identify stereo correspondences
Form 3D features
Invoke 3D feature editor for incremental position corrections
Compute attributes of the features
Display the accumulated features in all previous poses for the integration of the current pose
Establish pose-to-pose correspondence
Compute pose-transform relating current pose to first pose
A complete 3D model

Fig. 3. Division of responsibilities between the human operator and the computer.

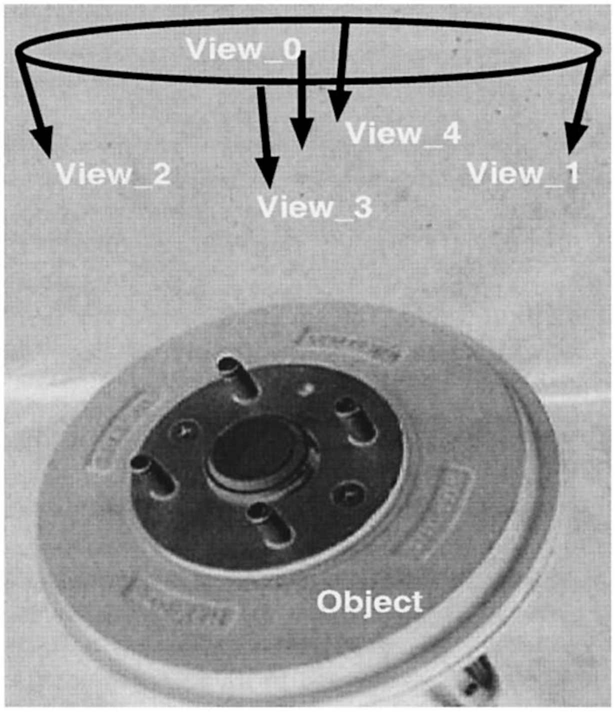


Fig. 4. Four viewpoints, View_0, View_1, View_2, View_3, and View_4, used for the imaging of an object in each stable pose.

- Apply the Canny edge operator to each image.
- Identify vertices if any are found.
- Load the resulting images into the model building program.

The reason for this initial row of buttons is to give the human a choice of low-level processing routines. Suppose we wanted to include other edge detectors in our system, to incorporate them into the computational process all we would have to do is create a button for each detector. Each button in this row is *modal*, meaning that each button can be clicked only after some other particular button has been clicked.

The buttons of the third row—these are the buttons with graphical icons on them—ask the user whether the feature to be extracted from an image is a polyhedral feature, an elliptical curved feature, a circular feature corresponding to cylindrical shape on the surface of the object, or a general curved feature. In the rest if this section, we will explain separately how the interaction proceeds for the polyhedral and the curved feature cases.

A. Human Interaction for Identifying Polyhedral Features

Extraction of a polyhedral feature begins with the human selecting those points in the image that correspond to the vertices of the object. The image used for this purpose is the edge-extracted version of a perpendicular view of the object. Before the human interacts with this image, the computer also applies a vertex detector to the output of the edge detector; the vertices are shown as dark filled circles in the image (Fig. 6). The view in which the human first identifies the features for a given pose of the object, the central view, is referred to as View_0 for that pose. The operation of the vertex detector is presented in greater detail in [17].

Since the vertex detector cannot be expected to work with 100% precision, the GUI gives the human the freedom to click anywhere he/she wants in order to define a new model vertex. However, if the point clicked on by the human is within three pixels of one of the system-supplied vertices, the system-supplied vertex is chosen. The three-pixel "slop" takes care of any fatigue-induced lack of precision in point clicking by the human. Additionally, three pixels translate roughly into 1.5 mm in the plane that corresponds to the base of the work area. This interval is roughly the best that our system can exhibit by way of spatial resolution in object reconstruction.

B. Human Interaction for Identifying Elliptical Features

As shown in Fig. 7, for elliptic shape extraction, a human is asked to select a representative set of points in View_0 along the perceived boundary of the elliptic shape. Although one can get away with only six points, for incorporating some noise immunity our system insists that the user click on at least eight points. A least-squares elliptical fit is then made to these points and projected back into the image.

C. Human Interaction for Stereo Correspondence

Identification of a feature in View_0 is followed by its localization in 3-D. For polyhedral features, this is done by establishing correspondences between the selected vertices in View_0 and the vertices in the four other views for every pose of the object. For curved features, the localization in 3-D is accomplished by first choosing a cardinal set of points on the analytically computed feature in View_0 and then establishing the stereo correspondents of those points in the other views.

So basic to the 3-D localization of both polyhedral and curved features is the identification of a corresponding pixel in View_i for a given pixel in View_0 for i = 1, 2, 3, 4. Since the finding of the corresponding pixels cannot be fully automated for reasons that are well documented in the computer vision literature,

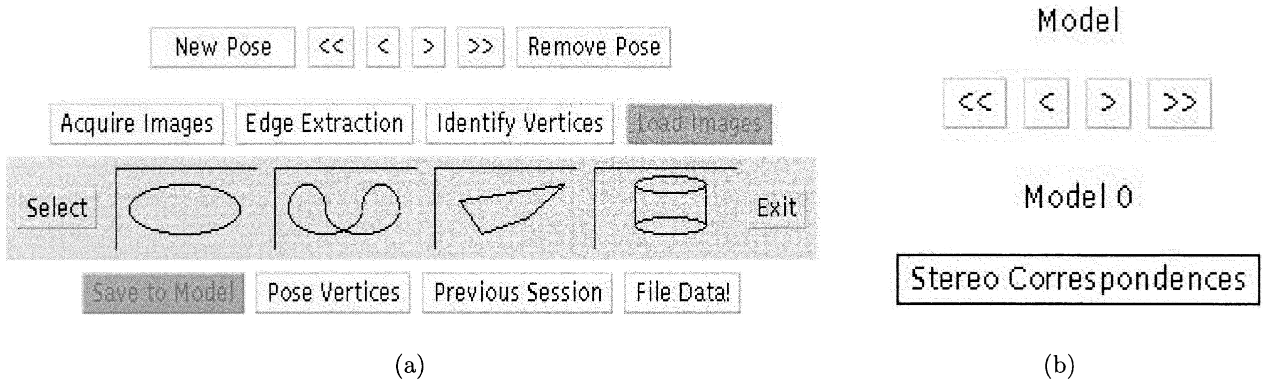


Fig. 5. Buttons and icons of the human-computer interaction editor.

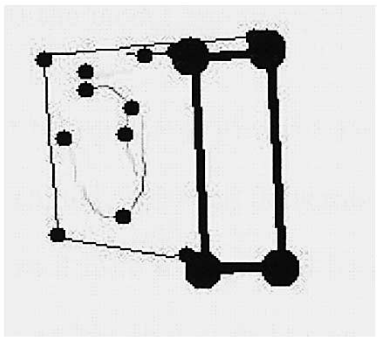


Fig. 6. Human selects vertices for polyhedral shapes directly on the top of View_0 image.

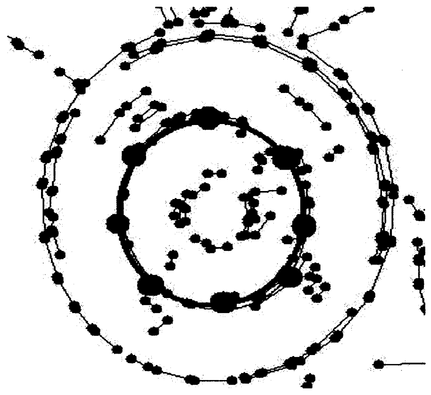


Fig. 7. Elliptical features superimposed on the edge image.

human's help is sought again. The human begins the process of helping out with stereo correspondence by clicking on the "Stereo Correspondences" button shown in Fig. 5(b), initiating a phase of interaction in which the different view images are shown sequentially in the GUI and the human asked to identify the corresponding pixel in each.

To speed up the human-assisted stereo correspondence in View_i, for each selected pixel in View_0 the system draws an epipolar line in View_i. The human then only has to click on what appears to be the best corresponding pixel on the epipolar line in View_i. To visually aid the human in this task, the GUI shows both the View_0 and the View_i images side by side; the View_0 pixel whose correspondent is sought is highlighted in the image on the right, as shown in Fig. 8, and the View_i image with the epipolar line on the left.

The stereo correspondences thus entered by the human are not taken on their face value. Even with the help provided by the system-drawn epipolar line, it is possible for a human to commit an error for an object with a large number of surface features. For example, shown at the bottom left of Fig. 8 is what the GUI shows for a curved object of moderate complexity in View_1 (The black arcs shown are the edge segments produced by low-level image processing routines.) Shown on the right is the pixel in View_0 whose correspondent the human must supply from the pixels displayed in View_1 on the left. As the reader can see, there are multiple competing pixels in View_1 on the epipolar line in the vicinity of where one would expect to find the corresponding pixel; the human must select one of those. Given this potential for human error, the system subjects

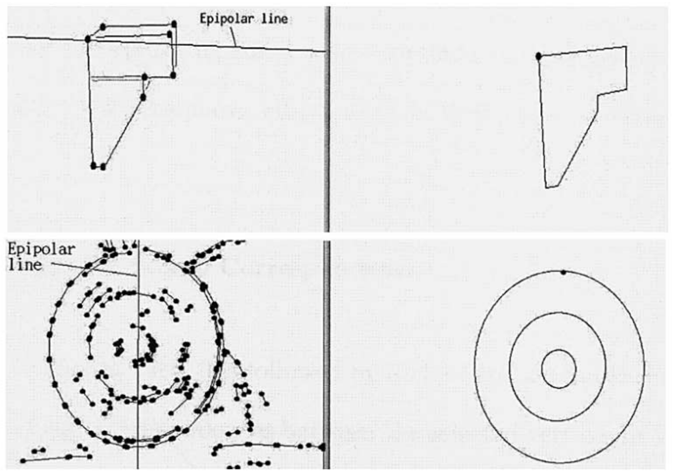


Fig. 8. Stereo correspondence of polyhedral and elliptical object. The image at top right shows the mid-level feature groupings extracted with human assistance from View_0. The computer highlights vertices in this partial model, one vertex at a time, and asks the human to identify a corresponding vertex in the View_i image at top left. To help the human, the computer draws the epipolar line in the left image as shown. The bottom two images show the same process for curved features.

the human-supplied correspondents to checking by all of the usual stereo constraints:

1) the epipolar constraint;
2) the ordering constraint;
3) the multiple view constraint.

Despite the fact that the epipolar line is drawn in the image to help the human, we still apply the epipolar constraint to the human-supplied points. This is to allow the human to select points that may not be exactly on the drawn epipolar line, but only in the general vicinity.

D. Human Interaction for Pose-to-Pose Correspondence

When an object is placed under the robot-mounted camera system, only those features can be acquired that are not occluded from the camera viewpoints used. In order to relate the features acquired from one pose with the features acquired from a previous pose, the system needs to know the pose transformation between the two poses by establishing feature correspondence between the two poses.

1) Pose-to-Pose Correspondence for Polyhedral Features: The human interaction protocol for establishing the pose-to-pose correspondences is designed in a way so as to mitigate the cognitive and perceptive burden on the human. As poses are integrated, we end with a growing partial model that can become perceptually complex toward the end of the pose sequence. As the partial model becomes complex, it can become difficult for a human to see which vertices in the current pose correspond to what vertices in the model accumulated so far. To keep the interaction efficient and to facilitate the visualization of such correspondences, the graphical user interface presents the following information to the human.

- Only the View_0 images are shown from the current pose and the latest pose already integrated into the partial model. The pose to pose correspondence is carried out for only the vertices visible in View_0 images in the different poses. The reader will recall that for the View_0 image, the camera is directly over the object. The other four images, indexed 1 though 4, provide stereo triangulation data for the vertices in View_0. It is possible that we may uncover additional 3-D triangulated vertices between the other four views, but usually such vertices will be visible in only two or three images, making their stereo correspondence not as robust as for the vertices in the View_0 image.
- Since the 3-D coordinates of the vertices are already known at this stage, the GUI also shows the wireframe representations of the reconstruction in the current pose and the reconstruction corresponding to the partial model. *The human is allowed to rotate in 3-D each of these reconstructions to better visualize how the current pose fits to the partial model constructed from the previous poses.*
- Note that the wireframe representations of the previous step are purely for the purpose of a human's visualization of the correspondences. The correspondences themselves must be entered by mouse clicking in the View_0 images for the current pose and one of the View_0 images for the poses already in the model.

Using these graphical devices, the human-computer interaction protocol for entering pose-to-pose correspondences is as follows.

1) The View_0 image of the current pose and the View_0 image of the latest pose in the accumulated partial model are shown side-by-side as in Fig. 9(a) and (b). The pose-to-pose vertex correspondences must be entered by clicking in one of these images as we will describe below. Buttons are provided in Fig. 5 for switching to the View_0 images of the other poses in the accumulated model if that's desired by the human. The need for doing so will become clear shortly.
2) To facilitate the visualization of current-pose to the accumulated-model correspondences, the wireframe representations for the 3-D reconstructions of the current pose and of the accumulated partial model are shown directly below the View_0 images, as in Fig. 9(c) and (d). As was mentioned before, these wireframes are meant to be rotated in 3-D.
3) The system highlights each vertex in the current pose View_0 image, as shown in Fig. 9(a), and asks the user to click on the corresponding vertex in the partial-model View_0 image on the right. As mentioned before, the human can switch to the other View_0 images in the partial model if the corresponding vertex is not visible in the one displayed.
4) Each vertex in the left image for which the human supplies a corresponding vertex in the right image is added to a database of object vertices for which 3-D coordinates can be calculated. Two 3-D vertices thus established are connected in a wireframe representation of the accumulated partial model if their 2-D projections are connected in any of the View_0 images included in the partial model so far.

If the human enters an incorrect correspondence between a vertex in the current pose and a vertex in the model already accumulated, the system discovers the inconsistency of the transformation by computing the error criterion that is presented in Section III-C. When an error is discovered, a message to that effect is flashed on the screen. The human then has the option of either entering new correspondences for the current pose or simply ignore the error message.

2) Pose-to-Pose Correspondence for Curved Features: While the main issue in establishing pose to pose correspondence for polyhedral shapes is the identification of corresponding vertices between two different poses, the main issue for curved features is the identification of the correct corresponding curves. Fig. 10 shows the View_0 images for a curved-feature object in the current pose in (a) and in (b) the View_0 image in the previous pose. Superimposed on the images in both (a) and (b) are the features extracted with human help in those two images. Shown in (c) is a wireframe reconstruction of the features extracted from (a). And, shown in (d) is the accumulated partial model before the current pose is integrated into it. The human can rotate the wireframe feature reconstruction in the current pose against the wireframe reconstruction of the accumulated model to get a better sense of how the current pose fits into the partial model.

As these figures illustrate, the main pose-to-pose correspondence problem for curved objects stems from possible confusion as to which curved feature in one pose goes with which curved

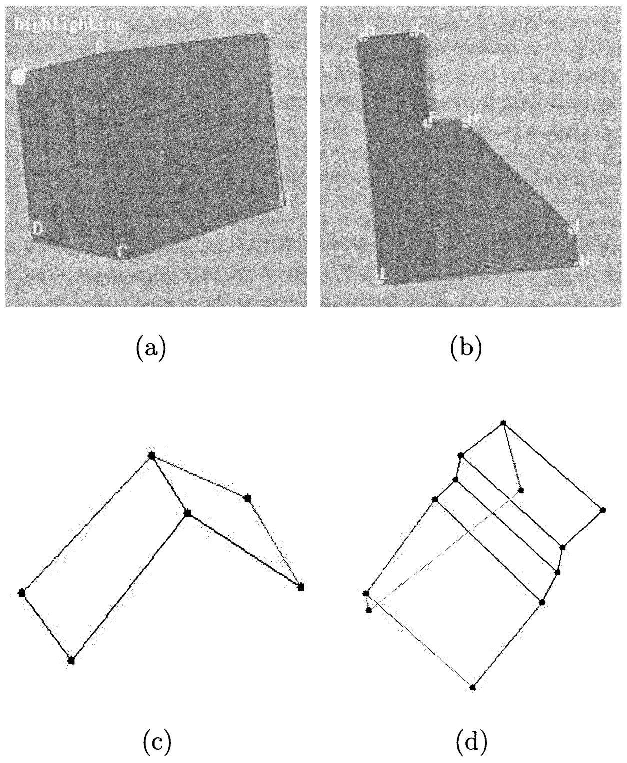


Fig. 9. For pose-to-pose correspondence, the View_0 image in the current pose is shown in (a), whereas (b) shows the View_0 image of the latest pose in the accumulated partial model. The user can switch to the other View_0 images in the accumulated model by clicking on a button. Shown in (c) is a wireframe representation of the partial model corresponding to the current pose of the object. Shown in (d) is the accumulated partial model using all of the previous poses of the object.

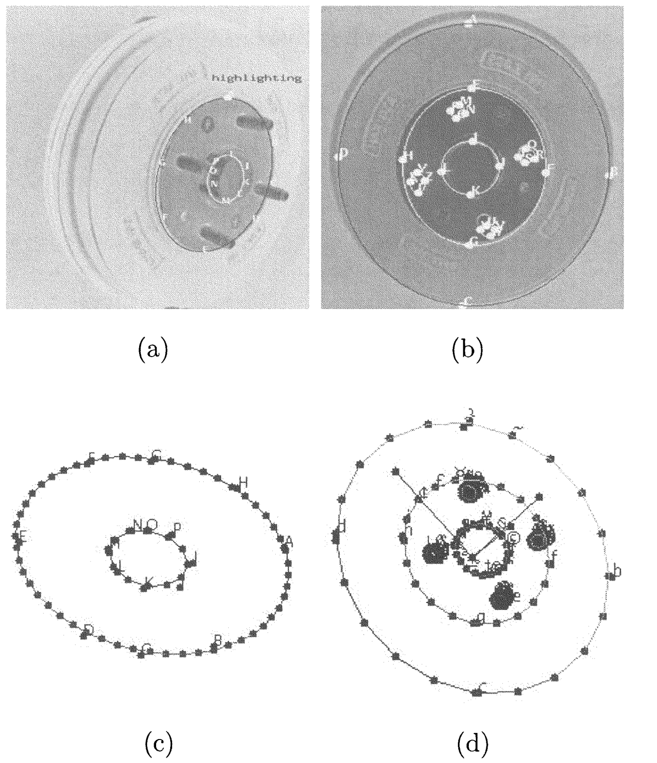


Fig. 10. For pose-to-pose correspondence, the View_0 image in the current pose is shown in (a), whereas (b) shows the View_0 image of the latest pose in the accumulated partial model. The user can switch to the other View_0 images in the accumulated model by clicking on a button. Shown in (c) is a wireframe representation of the partial model corresponding to the current pose of the object. Shown in (d) is the accumulated partial model using all of the previous poses of the object.

feature in another pose. The human interaction protocol here is simple: all that a human has to do is to mark for each curved feature in the View_0 image on the left its corresponding curved feature in the View_0 image on the right. And, as for the polyhedral case, the human can rotate the wireframe representations in 3-D before making the markings.

After establishing the curve-to-curve correspondences, the system upgrades its 3-D description of the curve taking into account the corresponding 2-D curves supplied by the two poses. The mathematics of how that is done will be explained later in Section IV.

III. Modeling Issues for Polyhedral Features

A. Calculation of the 3-D Coordinates of a Vertex Feature

As explained earlier in Section II, stereo correspondence for each vertex in a View_0 image is established by the system displaying in turn each of the other images, with the epipolar lines superimposed on each, and then asking the human to click on the correct corresponding vertex. The question then is how to optimally calculate the 3-D coordinates of the object point that gave rise to the vertices in the images.

Let i be the index for the five images collected in a given pose, and let C^i be the calibration matrix for the i^{th} viewpoint, the relationship between the (x,y,z) world coordinates of a scene vertex and the pixel coordinates (u_i, v_i) of the vertices in the $i = 0, 1, 2, 3, 4$ images is given by [14], [18]

$$\begin{bmatrix} C^i_{11} - u_i C^i_{31} & C^i_{12} - u_i C^i_{32} & C^i_{13} - u_i C^i_{33} \\ C^i_{21} - v_i C^i_{31} & C^i_{22} - v_i C^i_{32} & C^i_{23} - v_i C^i_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -C^i_{14} + u_i C^i_{34} \\ -C^i_{24} + v_i C^i_{34} \end{bmatrix} \tag{1}$$

where C^i_{mn} are elements of the calibration matrix C^i in the i-th $(i = 1, 2 \ldots)$ camera viewpoint (the viewpoint corresponding to the View_i image). From (1), the 3-D point $\boldsymbol{x} = (x, y, z)^T$ can be obtained as $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where

$$\boldsymbol{A} = \begin{bmatrix} C^1_{11} - u_1 C^1_{31} & C^1_{12} - u_1 C^1_{32} & C^1_{13} - u_1 C^1_{33} \\ C^1_{21} - v_1 C^1_{31} & C^1_{22} - v_1 C^1_{32} & C^1_{23} - v_1 C^1_{33} \\ C^2_{11} - u_2 C^2_{31} & C^2_{12} - u_2 C^2_{32} & C^2_{13} - u_2 C^2_{33} \\ C^2_{21} - v_2 C^2_{31} & C^2_{22} - v_2 C^2_{32} & C^2_{23} - v_2 C^2_{33} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ C^i_{11} - u_i C^i_{31} & C^i_{12} - u_i C^i_{32} & C^i_{13} - u_i C^i_{33} \\ C^i_{21} - v_i C^i_{31} & C^i_{22} - v_i C^i_{32} & C^i_{23} - v_i C^i_{33} \end{bmatrix}$$

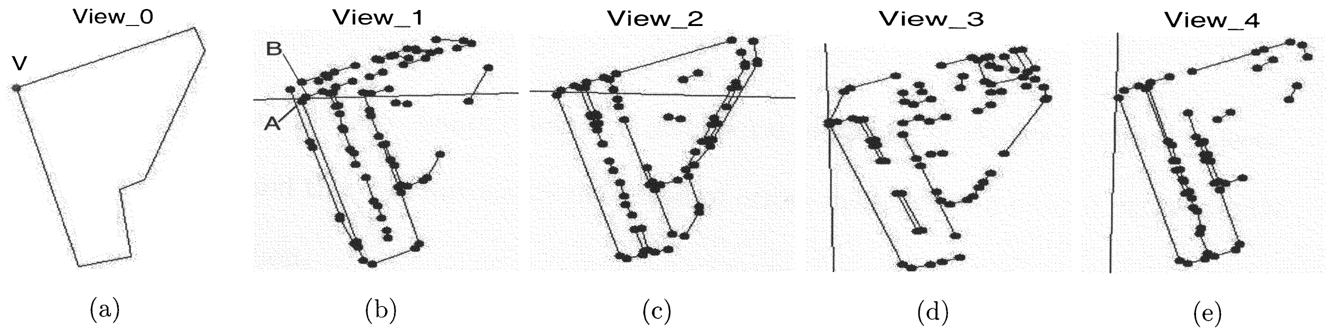


Fig. 11. Human interaction for establishing the stereo correspondences for the vertex marked V in the central viewpoint image in (a). The human is asked to click in each of the other four images to identify the corresponding points. The correct correspondent in (b) is marked A, but the human clicked on the point marked B.

$$\boldsymbol{b} = \begin{bmatrix} -C^1_{14} + u_1 C^1_{34} \\ -C^1_{24} + v_1 C^1_{34} \\ -C^2_{14} + u_2 C^2_{34} \\ -C^2_{24} + v_2 C^2_{34} \\ \cdots \\ \cdots \\ -C^i_{14} + u_i C^i_{34} \\ -C^i_{24} + v_i C^i_{34} \end{bmatrix}. \quad (2)$$

The least-squares estimate of $\boldsymbol{x} = (x, y, z)^T$ from the above equation is given by

$$\boldsymbol{x} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{b}. \quad (3)$$

It goes without saying that the accuracy in computing (x, y, z) depends essentially on the accuracy of the camera calibration matrix for each viewpoint. Our calibration procedures, discussed in [18], are based on the pinhole model of a camera with lens distortion.

B. Error Detection in Human Supplied Stereo Correspondences

We must allow for the possibility that the human might misclick on some of the points when identifying corresponding pixels among the five images. If the mouse click is way off the mark in one or more of the View_1 through View_4 images, the entire set of correspondences would be rejected for a given vertex in View_0 image by our thresholding logic that we will explain next.

We apply the following two tests to the human-supplied correspondences in all the stereo pairs we can form from the five images recorded in a given pose.

- Check if the epipolar constraint is satisfied by all five corresponding pixels in seven out of ten stereo pairs formed by the five images.
- Apply the ordering constraint to the supplied correspondences.

If the mouse click is way off the mark in one image out of five, in general one or both of these constraints would not be satisfied in four out of ten stereo pairs. Our acceptance threshold is 7, meaning that these two constraints should be satisfied in seven out of the ten stereo pairs we can form from the five images.

If the human has misclicked in one or more images and the above stated seven-out-of-ten rule is violated, we have 2 options.

TABLE I
ERROR IN AVERAGED DISTANCE USING FOUR IMAGES
(LEAVE-ONE-OUT IMAGE)

Image Indexed View	Error Distance (mm)
b,c,d,e (a-out)	8.4
a,c,d,e (b-out)	1.1
a,b,d,e (c-out)	6.0
a,b,c,e (d-out)	6.4
a,b,c,d (e-out)	7.2

Option 1: Reject the entire set of five corresponding pixels entered by the human and repeat the process.

Option 2: Identify the incorrect mouse click and have the human fix the error in just that one image.

We believe that the first option places an excessive burden on the human operator. We have therefore gone with the second option. In this option, the image in which the human misclicked is identified by computing the Euclidean distance $||\boldsymbol{Ax} - \boldsymbol{b}||^2$ on a one-leave-out basis, where $\boldsymbol{A}$ and $\boldsymbol{b}$ are as defined in (2). The leave-one-out basis here means that we compute this distance for $\boldsymbol{A}$, $\boldsymbol{x}$, and $\boldsymbol{b}$ as determined for each grouping of four out of five images. The set of four images that yields the smallest value for this distance identifies by elimination the image in which the human mis-clicked.

This error detection procedure will be illustrated with the help of Fig. 11 where we have shown the five images taken from the five viewpoints. Let's assume that the human has mis-clicked in the image labeled b. The correct correspondence in this image is marked A, but the human has clicked at the point marked B. This causes a failure of the pairwise epipolar constraint to be satisfied for the image pairs (b, a), (b, c), (b, d), (b, e). This means that the epipolar constraint is satisfied in only five out of the ten stereo pairs we can form from the 5 images—which is below our acceptance threshold of 7 out of 10. This violation of the acceptance threshold then causes the computation of the Euclidean distance $||\boldsymbol{Ax} - \boldsymbol{b}||^2$ to be carried out for the identification of the image that is the source of a bad correspondence. Shown in Table I is this distance for the five images of Fig. 11. As is clear from this example table, the image labeled b is the source of the bad correspondence.

When an error in the human-supplied correspondences is detected by our seven-out-of-ten rule, the human is alerted by flashing a message on the screen. The human has the choice of ignoring the message. If the message is not ignored by pressing on the "ok" button, the system proceeds with the computation of the distance $||\boldsymbol{Ax}-\boldsymbol{b}||^2$ for identifying the source of the problem. The human can subsequently re-supply a more correct choice.

C. Accumulating and Merging Features From Different Poses

As was mentioned before, in the learning phase the object is placed in different poses under the camera system so that all of its surfaces can be captured in the model. In each pose, the multiple-view vision system acquires a fresh set of features on the object and it is the accumulation of all the features from all the poses that lead to the final model. In order to relate the features acquired from one pose with the features acquired from a previous pose, the system needs to know the pose transformation between the two poses. In other words, the system needs to know the rotational matrix and the translational vector that would take the object in its current pose under the camera system to the pose that was used previously.

The process of integrating all the 3-D coordinates collected from the five different views in one pose with all the 3-D coordinates from all previous poses will be referred to as *model accretion*. Model accretion requires that there be some common features visible to the camera system between the current pose and the partial model accumulated from all previous poses. By mouse clicks, the human in the loop needs to tell the system what vertices acquired in one pose correspond to which vertices in a previous pose. As the discussion in the rest of this section will reveal, the human must identify at least three different noncoplanar vertices that are common between the current pose and previously accumulated partial model for the system to be able to carry out their integration. Of course, in practice, if more than three noncoplanar points are available, the integration will be that much more robust.

Fundamental to model accretion is the calculation of the rotation matrix and the translation vector that would take the object from the new pose into its old pose. It is through the rotation matrix and the translation vector, that the system knows where to place the newly acquired vertices and other features in relation to the previously acquired vertices and features. We adopt the quaternion approach [5] for the optimum calculation of the rotation matrix and the translation vector.

The coordinate frame associated with the first pose is used to accumulate the information gleaned from all other poses. So when the second pose becomes available, the system first calculates the rotation and translation $(\boldsymbol{R}, \boldsymbol{t})$ from the common vertices between the new pose and the first pose and then adds the new vertices seen the second pose to the integration taking place the coordinate space corresponding to the first pose.

An important engineering question that pertains to this integration of poses is how does the system discover the error if the human enters an incorrect correspondence between a vertex in the current pose and a vertex in the model already accumulated? We use the following two criteria to detect such errors.

1) Given a set of vertices that are common to two poses, let's call them Pose 0 and Pose 1, we use the quaternion approach to find the optimum rotation matrix between the two poses. So given the vertices $v_0^0, v_1^0, \ldots .v_n^0$ as the orientation vectors associated with the vertices chosen in Pose 0 and $v_0^1, v_1^1, \ldots .v_n^1$ as the corresponding orientation vectors in Pose 1, the quaternion approach consists of first establishing a matrix (4) shown at the bottom of the page, and solving an eigenvalue problem for the following vector-matrix equation:

$$\left(\sum_{i=0}^{n} \boldsymbol{A_i} \cdot \boldsymbol{A_i^T}\right) \cdot \boldsymbol{Q} = \lambda \boldsymbol{Q}. \tag{5}$$

The eigenvalues and the eigenvectors of the matrix on the left can be obtained by carrying out the following minimization:

$$minE_{\boldsymbol{R}}^2 = arg\min_{\boldsymbol{Q}} \boldsymbol{Q^T} \cdot \left(\sum_{i=0}^{n} \boldsymbol{A_i} \cdot \boldsymbol{A_i^T}\right) \cdot \boldsymbol{Q}. \tag{6}$$

The eigenvector that we choose as a solution for the rotation between the two poses should correspond to the smallest eigenvalue. It can easily be shown that the fitting error for computing the best rotation matrix in this way equals the magnitude of the smallest eigenvalue [3], [5], [13]. To see this, we first note that the rank of the matrix in our vector-matrix equation is four. Let's represent the four eigenvalues by λ_0, λ_1, λ_2, λ_3, where $|\lambda_0| \leq |\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$, and the associated normalized eigenvectors by Q_0, Q_1, Q_2, and Q_3. The mean squared error corresponding to each eigenvector can be obtained by substituting (5) in (6)

$$minE_{\boldsymbol{R}}^2 = \boldsymbol{Q_0}^T \lambda_0 \boldsymbol{Q_0} = |\lambda_0|. \tag{7}$$

Obviously, by applying a decision threshold to the smallest eigenvalue, we can either accept or reject a calculated rotation matrix $(\boldsymbol{R})$. We have observed experimentally that we can use the same threshold for all poses and for all objects in our current library. The value of this decision threshold is given by 1.0^{-3}.

Fig. 9 shows the case in which the 3-D partial model constructed in Pose 1 was rejected on the basis of this criterion applied to the Pose 1 to Pose 0 rotation quaternion. The value of the smallest eigenvalue in this case was 0.0022.

2) The above criterion only looks at the errors in the calculation of the optimum rotation matrix. We also need to check the

$$\boldsymbol{A_i} = \begin{bmatrix} 0 & \left(\boldsymbol{v_i^1} - \boldsymbol{v_i^0}\right)_{\boldsymbol{x}} & \left(\boldsymbol{v_i^1} - \boldsymbol{v_i^0}\right)_{\boldsymbol{y}} & \left(\boldsymbol{v_i^1} - \boldsymbol{v_i^0}\right)_{\boldsymbol{z}} \\ -\left(\boldsymbol{v_i^1} - \boldsymbol{v_i^0}\right)_{\boldsymbol{x}} & 0 & \left(\boldsymbol{v_i^1} + \boldsymbol{v_i^0}\right)_{\boldsymbol{z}} & -\left(\boldsymbol{v_i^1} + \boldsymbol{v_i^0}\right)_{\boldsymbol{y}} \\ -\left(\boldsymbol{v_i^1} - \boldsymbol{v_i^0}\right)_{\boldsymbol{y}} & -\left(\boldsymbol{v_i^1} + \boldsymbol{v_i^0}\right)_{\boldsymbol{z}} & 0 & \left(\boldsymbol{v_i^1} + \boldsymbol{v_i^0}\right)_{\boldsymbol{x}} \\ -\left(\boldsymbol{v_i^1} - \boldsymbol{v_i^0}\right)_{\boldsymbol{z}} & \left(\boldsymbol{v_i^1} + \boldsymbol{v_i^0}\right)_{\boldsymbol{y}} & -\left(\boldsymbol{v_i^1} + \boldsymbol{v_i^0}\right)_{\boldsymbol{x}} & 0 \end{bmatrix} \tag{4}$$

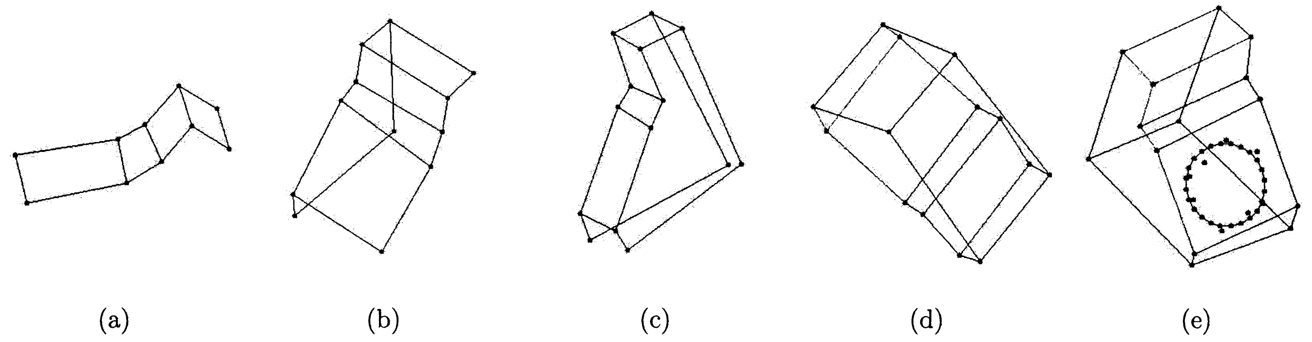


Fig. 12. Shown in this figure is the model accretion process for which the final result was shown in Fig. 1. The five images collected in the first pose yield the wireframe shown in (a). When this partial model is integrated with the data gleaned from the images in the second pose, we get the accumulated partial model in (b). This accumulation continues with the incorporation of each new pose, until we get the final wireframe shown in (e).

contribution made by translation errors in order to form a true estimate of the pose transform error. An integrated measure of error is given by $\sum_{i=0}^{n} |\boldsymbol{R}\boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{k}} + \boldsymbol{t} - \boldsymbol{p}_{\boldsymbol{i}}^{0}|^2/n$, where $(\boldsymbol{R}, \boldsymbol{t})$ are the computed pose of Pose 1 vis-a-vis Pose 0, and p_i^k and p_i^0 are the corresponding vertices for Pose k and Pose 0.

This composite measure of error gives an average value of the error on a per vertex basis. The per vertex basis allows for a single threshold to be used for either accepting or rejecting a calculated pose. If the user enters an incorrect vertex in Pose 1 as a correspondent for a given vertex in Pose 0, we can expect that the overall $\boldsymbol{R}$ and $\boldsymbol{t}$ for Pose k will be wrong and that this fact would be reflected in an unacceptable value for the above error measure. By trial and error, we have found that a decision threshold value of $1.0^{-2}\ m^2$ works well for the above criterion for our system.

An example of model accretion is shown in Fig. 12 which yields the final result of Fig. 1. The partial model constructed from the five images for the first pose is shown in (a). When this is integrated with the partial model extracted in the second pose, we get the accumulated partial model of (b). Further integration with the partial model extracted in the third pose yields (c), etc.

IV. Issues in Curved Shape Modeling

The previous section discussed issues related to polyhedral modeling of shapes or entire objects. In particular, we focused there on automatic detection of errors made when a human inadvertently enters wrong information.

In this section, we will do the same for curved object. But we must state at the very outset that we deal with a very limited case of curved shapes.[1] Our focus will be limited to planar elliptical features, as exemplified by the curved silhouettes on the industrial object shown in Fig. 10.

Lacking distinguishing vertices, curved features obviously cannot be extracted in a manner similar to polyhedral features. For a curved feature, the low-level image processing routines will usually output edge fragments at the curve boundaries, as shown by a processed image in Fig. 7. Given this kind of output from the low-level routines, we are faced by the following goals.

- The human must first indicate his/her intention to the computer that he/she is about to delineate a curved feature. Using menu-driven interaction, the human can also inform the computer as to what shape the human is about to delineate. Given the current scope of our system, these shapes will only be elliptical, which includes circular.
- The human must then click on a sufficiently large set of points on the edge-extracted portions of the boundary of the curved feature. Again, since we are currently limited to elliptical features, the user must click on eight points. Note that theoretically only six points are needed for a least-squares fit of an ellipse to the supplied points.
- Given 2-D ellipses extracted from the different camera images as projections of the same curved feature on an object, the system must then come up with an optimum 3-D description of the curved feature on the object.

In the rest of this section, we will address the computational issues involved in this interaction. Our discussion to follow will break the interaction into a sequence of four steps.

A. *Computational Steps for 3-D Modeling of Elliptical Features*

Step 1) The human clicks at eight points (u_i, v_i) for $i = 1, 2, \ldots 8$ on what the human sees as an elliptical feature in the central image for a given pose of the object. This is the image for View_0. An example of what a human clicks on for delineating elliptical features is shown in Fig. 7. A least-squares minimization algorithm [6] is used to fit an optimum 2-D ellipse to the points supplied by the human. The human-supplied pixel coordinates (u_i, v_i) are expressed in the form of a vector $\boldsymbol{x_i} = (u_i^2, u_i v_i, v_i^2, u_i, v_i, 1)^T$ for $i = 1, \ldots 8$ for each of the eight points selected by the human. The ellipse parameters are then obtained by a least-squares solution of $\boldsymbol{a} = (a, b, c, d, e, f)^T$ for an ellipse of description $F(\boldsymbol{a}, \boldsymbol{x_i}) = \boldsymbol{a} \cdot \boldsymbol{x_i} = 0$, which is the same as optimally solving $au_i^2 + bu_iv_i + cv_i^2 + du_i + ev_i + f = 0$.

Step 2) After delineating the ellipse in the central image, we need to find the "corresponding" ellipses in the other four images of the five images recorded in each pose of the object. For this purpose, the computer first chooses its own eight points on the analytically derived ellipse from the View_0 image in Step 1.

[1]*Nevertheless, the curved shapes we are able to handle describe a large class of objects with curved features in industry.*

These points are the cardinal points on the ellipse, cardinal in the sense that they are located at the intersections of the major and minor axes with the elliptical boundary and of the diagonal lines in-between. The central ellipse in the top portion of Fig. 13 represents the ellipse extracted from the View_0 image. The cardinal points on this ellipse as positioned by the computer are labeled A, B, C, D, E, F, G, and H.

For each of the above cardinal points on the ellipse in the View_0 image, the computer draws an epipolar line in each of the other four views. As with the polyhedral case, this is merely to help the human identify the pixels corresponding to the View_0 cardinal pixels in the other four views for a given pose. The human then clicks on what he/she considers to be the correct corresponding pixel in each of the four views presented in a sequence in the same manner as for the polyhedral case. As shown in the left ellipse in the top portion of Fig. 13, the computer draws an epipolar line in the image and the human must then select a point in the vicinity of the intersection of the epipolar line and the perceived ellipse in that image.

Step 3) From the pixel coordinates of all the corresponding pixels for a given cardinal point in the View_0 image, the system then computes the 3-D (x, y, z) coordinates of that point on the object.

Step 4) Given 3-D coordinates of all eight cardinal points on the ellipse as located by the human in the View_0 image, the system now fits an optimum planar ellipse to these eight 3-D points using the following two steps: 1) First a least-squares plane is fit to the points in 3-D world coordinates, as illustrated in Fig. 13 Step 4:). 2) The eight cardinal points are then projected onto this best-fitting plane.

A planar 3-D ellipse calculated in this manner is then characterized by the following parameters.

1) The two parameters that describe the orientation of the optimum plane through the eight points.
2) The center of the mass of the eight points as projected onto this plane. This center of mass, which is also the center of the ellipse, will be represented by (X_c, Y_c).
3) The focal lengths f_a and f_b of the ellipse in the best-fitting plane are then calculated by using standard formulas [8], as is also the orientation θ of the major axis of the ellipse with respect to the XY plane of the world coordinates.

To estimate the focal length and the orientation of the plane in the best-fitting 2-D plane, a new 2-D coordinate frame is established in the best-fitting plane, with its center at the point (X_c, Y_c) and with its X and Y axes in two arbitrary but mutually perpendicular directions. A 4 $\times$ 3 homogeneous transformation matrix keeps track of the orientations of the X and Y directions in the best-fitting plane vis-a-vis the world coordinates.

B. Pose-to-Pose Integration of Curved Features

Recall that the goal of pose-to-pose integration of information is to find a reliable transformation matrix between two different poses so that the information gathered for two different poses can be represented in a single coordinate frame. For the polyhedral case, this was achieved by identifying corresponding vertices in two different poses.

For the curved feature case a strategy similar to that used for the polyhedral case is used to find the pose-to-pose correspondence. The quaternion pose calculation formulas are used on the eight cardinal points on the same ellipse in the two different poses. These formulas then directly yield the pose transform.

From the standpoint of the human interaction involved, the main focus of pose-to-pose merging process is for the human to tell the computer which ellipse in one pose corresponds to which ellipse in a second pose. Our interaction editor makes this possible by asking the human to click on the correct corresponding ellipse in Pose 2 for every ellipse highlighted in Pose 1.

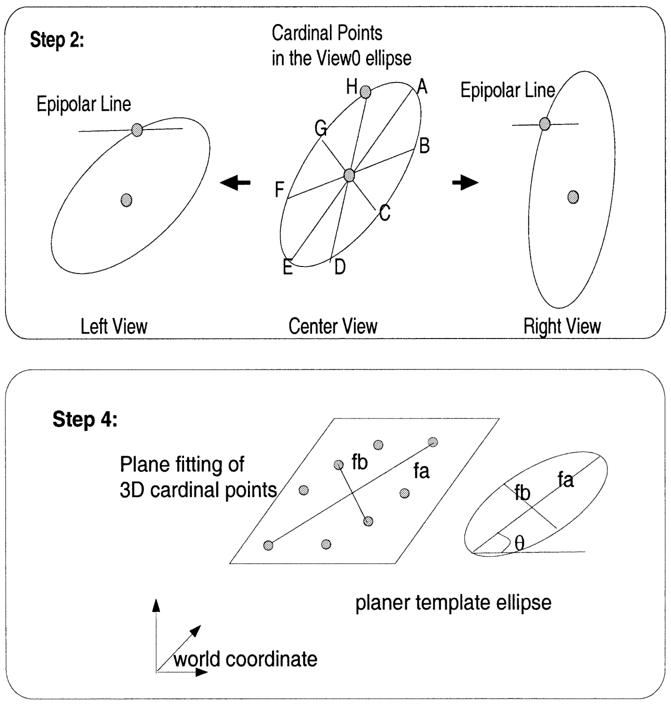


Fig. 13. Step 2: Compute epipolar lines on the representative points. Step 4: A planer fitting in the cardinal points.

V. Model Accuracy Evaluation

The results of an evaluation study presented here show the accuracy with which our system can construct models of new objects. But before presenting the evaluation results, we will first describe our system configuration in the next subsection.

A. System Configuration

The Graphics Editor that allows a human to interact with the images in the manner described in this paper is written in Java. The computer vision routines for low and mid and high level processing are all in C/C++. Java calls up the C functions as needed and retrieves their output for display when so necessary. The system is installed on a SUN Workstation ULTRA 10. It usually takes several minutes for a human working interactively with the GUI to generate an object model.

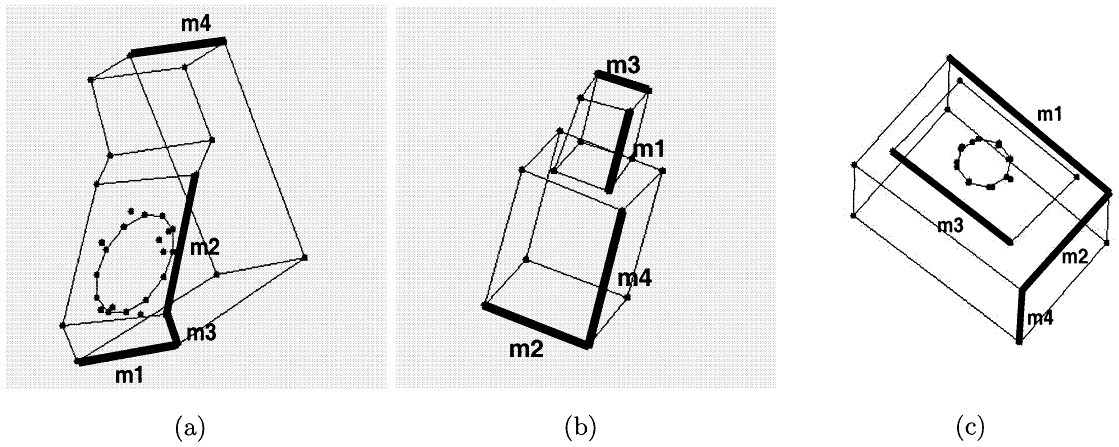


Fig. 14. Acquired 3-D object models of polyhedral objects with measurement labels m1–m4 corresponding in Table II.

TABLE II
THIS TABLE SHOWS THE ACCURACY OF OUR HUMAN-ASSISTED MODEL CONSTRUCTION SYSTEM FOR THREE OBJECTS THAT ARE PRIMARILY POLYHEDRAL. ALL DIMENSIONS ARE IN MILLIMETERS

Object	m1:acquired/true	m2	m3	m4
Object (a)	90/87	77/74	37/36	86/87
Object (b)	60/63	78/75	36/37	102/100
Object (c)	97/100	66/69	22/24	35/36

TABLE III
PERCENTAGE ERROR FOR THE ACCURACY RESULTS SHOWN IN TABLE II

Object	Max Error(%)	Min Error(%)
Object (a)	5.0	4.4
Object (b)	2.9	2.2
Object (c)	8.3	3.0

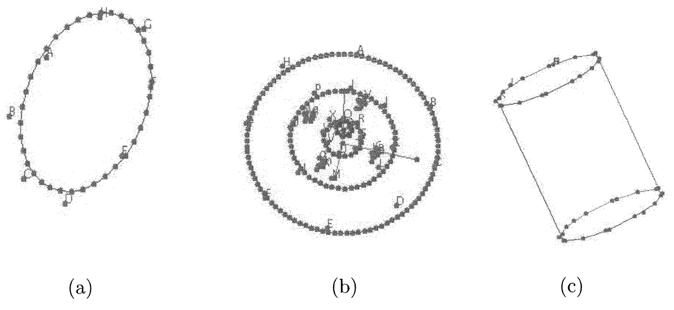


Fig. 15. Acquired 3-D object models of curved shape objects.

The imaging data is acquired by a monocular camera mounted on a robotic wrist. For each pose of the object, five images are taken by moving the robotic effector to different viewpoints. The robotic-wrist-mounted robotic vision system consists of a Sony *DC*-47 monocular 1/3 *inch* CCD camera with a Pulnix Lens of focal-length 16 *mm*. The robot end-effector automatically moves to the different viewpoints to generate all the multiple views needed. The acquired images are digitized on a 512 $\times$ 480 array of pixels.

TABLE IV
NUMERATOR IN EACH ENTRY IS THE VALUE IN A RECONSTRUCTED MODEL AND THE DENOMINATOR THE GROUND TRUTH

Object	$foc1_x$:acquired/true	$foc1_y$	$foc2_x$	$foc2_y$	foc_θ
Object (a)	99/102	1/2	213/204	7/2	149/141
Object (b)	3/1	5/1	3/1	5/1	131/136
Object (c)	7/12	2/6	4/6	1/8	37/38

TABLE V
PERCENTAGE ERRORS IN THE FIVE PARAMETERS OF THE RECONSTRUCTED MODELS

Object	Max Error(%)	Min Error(%)
Object (a)	4.9	2.3
Object (b)	7.1	2.0
Object (c)	6.6	2.2

B. Results on Polyhedral Objects

We will show accuracy results on the three polyhedral objects whose wireframe images are shown in Fig. 14. The accuracies will be shown with respect to the measurements marked m1, m2, m3 and m4 in the figure. For each of these objects, Table II shows the true values of these measurements and the values in the constructed models in a typical run of our system.

Table III presents the above accuracy results in the form of error percentages.

C. Results of Curved Objects

This section shows model reconstruction accuracy results for curved objects, meaning objects with planar elliptical features.

The accuracy of model construction is evaluated on the basis of the values of the coordinates of the two foci of a planar ellipse and the orientation θ of the best fitting plane to the ellipse in the world coordinates. In the best fitting plane, as defined in Section IV, the coordinates of the foci are measured with respect to a 2-D coordinate frame that is centered at the center of the ellipse, with its X axis lined up with the major axis and the Y axis lined up with the minor axis. If we denote the two foci by $foc1$ and $foc2$, their coordinates in the best fitting plane can be represented by $(foc1_x, foc1_y)$ and $(foc2_x, foc2_y)$.

We used three curved shape objects for our evaluation study. Fig. 15 shows pictorially the wireframe representations of the corresponding models constructed by our system. The object in (a) is a very thin and flat elliptical shaped aluminum plate, the object in (b) the same as shown earlier in Fig. 10, and the object in (c) a wooden cylinder.

Table IV summarizes these results for each object in the evaluation study.

Table V shows the accuracies obtained for the three objects in the form of percentages of the ground-truth values.

VI. Conclusion

Our human-assisted model reconstruction system calls for an object to be placed in all its stable poses on a work table under a robot-controlled camera. In each stable pose, one central and four peripheral images of the object are taken. Low level image processing routines are applied to all such images (no human intervention here) and subsequently made available for human interaction to construct 3-D models.

The nature of human interaction depends on whether a feature to be extracted is polyhedral or planar curved shape. For polyhedral features, the human first helps identify each planar face by clicking on the vertices and then by identifying corresponding vertices in the stereo images. In the latter task, the computer helps the human by drawing epipolar lines in the peripheral view images where stereo correspondences are sought. This human input helps the computer figure out the 3-D coordinates of the vertex features in each stable pose of the object. Subsequently, the human is asked to help out with establishing pose-to-pose correspondence by identifying vertices common to the different poses. The protocol for the curved features is different: the human first clicks on 8 points in the central image to help the computer fit an elliptical curve to the points. The computer then identifies eight cardinal points on the ellipse thus constructed in the central image. Subsequently, the computer seeks from the human the correspondents of these eight cardinal points in the four peripheral images; the computer helps the human in this task by drawing epipolar lines in the peripheral images, as was done for the polyhedral case. Thus the computer constructs a 3-D model of a planar elliptical feature. For pose to pose integration the human identifies for the computer which ellipses in one pose correspond to that ellipse in a different pose.

Our accuracy results bear out the feasibility of our human-computer interaction protocols. Further extensions of our work would include general 3-D surface modeling.

Acknowledgment

The authors dedicate this paper *in memoriuam* to Pamela Wilson who programmed the 1st Java implementation of the 3–D Graphics Editor that was instrumental in starting the project. They are also grateful to F. Maslar, V. Bolhouse, and the Staff of the Advanced Manufacturing Technology Department, Ford Company, for providing constant feedback on the appropriateness of various human-computer interaction strategies experimented on over the course of the project. The work reported here would not be possible without the help of many of the past and present members of this group, in particular, A. Kosaka, G. DeSouza, and E. Suryadi.

References

[1] R. C. Bolles and P. Horaud, "3-DPO: a three-dimensional part orientation system," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 3–26, 1986.
[2] Canoma, MetaCreation.*http://www.canoma.com* [Online]
[3] C. Chen and A. Kak, "A robot vision system for recognizing 3-D objects in lower-order polynomial time," *IEEE Trans. Syst. Man. Cybern.*, vol. 19, pp. 1535–1563, Nov./Dec. 1989.
[4] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *Proc. SIGGRAPH'96 Conf.*, Aug. 1996, pp. 11–20.
[5] O. D. Faugeras, *Three-Dimensional Computer Vision*. Cambridge, MA: MIT Press, 1993.
[6] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipse," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, pp. 476–480, May 1992.
[7] A. R. J. Franois and G. G. Medioni, "Interactive 3-D model extraction from a single image," *Image Vis. Comput.*, vol. 19, no. 6, pp. 317–328, Apr. 2001.
[8] G. Fuller, *Analytic Geometry*. Reading, MA: Addison-Wesley, 1973.
[9] L. Grewe and A. C. Kak, "Interactive learning of a multi-attribute hash table classifier for fast object recognition," *Comput. Vis. Image Understanding*, vol. 61, no. 3, pp. 387–416, 1995.
[10] S. Heuel and R. Nevatia, "Including interaction in an automated modeling system," in *Proc. Int. Symp. Comput. Vis.*, 1995, pp. 383–388.
[11] K. Ikeuchi and T. Suehiro, "Toward and assembly plan from observation part I: task recognition with polyhedral objects," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 368–384, June 1994.
[12] ImageModeler, RealViz.*http://www.realviz.com* [Online]
[13] W. Y. Kim and A. C. Kak, "3-D object recognition using bipartite matching embedded in discrete relaxation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 224–251, Mar. 1991.
[14] A. Kosaka and A. C. Kak, "Stereo vision for industrial applications," in *Handbook of Industrial Robotics*, S. Y. Nof, Ed. New York: Wiley, 1999, pp. 269–294.
[15] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 799–822, Dec. 1994.
[16] M. Levoy, K. Pulli, B. Curless, Z. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, "The digital Michelangelo project: 3-D scanning of large statues," in *Proc. SIGGRAPH'00*, 2000, pp. 131–144.
[17] Y. Motai, Ph.D dissertation, School of Elect. Comput. Eng., Purdue Univ., West Lafayette, IN, Aug. 2002.
[18] Y. Motai and A. Kosaka, "SmartView: hand-eye robotic calibration for active viewpoint generation and object grasping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2001, pp. 2183–2190.
[19] K. Nishino, Y. Sato, and K. Ikeuchi, "Appearance compression and synthesis based on 3-D model for mixed reality," in *Proc. Int. Conf. Comput. Vis.*, 1999, pp. 38–45.
[20] K. Ogawara, S. Iba, T. Tanuki, H. Kimura, and K. Ikeuchi, "Acquiring hand-action models by attention point analysis," in *Proc. Int. Conf. Robot. Automat.*, vol. 1, 2001, pp. 465–470.
[21] PhotoModeler Pro, Eos Systems Inc. *http://www.photomodeler.com* [Online]

Yuichi Motai (M'03) received the B.E. degree in instrumentation engineering from Keio University, Japan, in 1991, the M.E. degree in applied systems science from Kyoto University, Japan, in 1993, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 2002.

The work presented in this paper was conducted while he was with the Robot Vision Laboratory, Purdue University.

Dr. Motai is a member of ACM.

Avinash Kak is a Professor of electrical and computer engineering at Purdue University, West Lafayette, IN. He is most recently the author of the book *Programming with Objects: A Comparative Presentation of Object Oriented Programming with C++ and Java* (New York: Wiley, 2003).