Robot Acting on Moving Bodies (RAMBO): Interaction with Tumbling Objects

Larry S. Davis, Daniel DeMenthon, Thor Bestul, Sotirios Ziavras, H.V.Srinivasan, Madhu Siddalingaiah, and David Harwood

Computer Vision Laboratory
Center for Automation Research
University of Maryland, College Park, MD 20742

Abstract

Interaction with tumbling objects will become more common as human activities in space expand. Attempting to interact with a large complex object translating and rotating in space, a human operator using only his visual and mental capacities may not be able to estimate the object motion, plan actions or control those actions.

We are developing a robot system (RAMBO) equipped with a camera, which, given a sequence of simple tasks, can perform these tasks on a tumbling object. RAMBO is given a complete geometric model of the object. A low level vision module extracts and groups characteristic features in images of the object. The positions of the object are determined in a sequence of images, and a motion estimate of the object is obtained. This motion estimate is used to plan trajectories of the robot tool to relative locations nearby the object sufficient for achieving the tasks.

More specifically, low level vision uses parallel algorithms for image enhancement by symmetric nearest neighbor filtering, edge detection by local gradient operators, and corner extraction by "sector filtering". The object pose estimation is a Hough transform method accumulating position hypotheses obtained by matching triples of image features (corners) to triples of model features. To maximize computing speed, the estimate of the position in space of a triple of features is obtained by decomposing its perspective view into a product of rotations and a scaled orthographic projection. This allows us to make use of 2D lookup tables at each stage of the decomposition. The position hypotheses for each possible match of model feature triples and image feature triples are calculated in parallel. Trajectory planning combines heuristic and dynamic programming techniques. Then trajectories are created using dynamic interpolations between initial and goal trajectories. All the parallel algorithms run on a Connection Machine CM-2 with 16K processors.

1 Introduction

The problem of robotic visual navigation has received considerable attention in recent years, but research has mostly concentrated on operations in static environments [1–11]. The area of robotics in the presence of moving bodies has seen little activity so far [12–16]. We are developing a control system which should allow a robot with a camera to accomplish a sequence of actions on a moving object. The development effort most related to our work is taking place at the Jet Propulsion Laboratory [16].

One primary application for this type of research could be the development of an autonomous vehicle able to develop strategies for intercepting a moving target on the ground such as another vehicle. But our approach seems general enough to be applied to other domains such as robotics in space. For example, a robotic arm building a structure in the absence of gravity might require the capability of interacting autonomously with moving objects. During a teleoperated assembling process one of the building elements could break loose from the gripper. Then the natural motion of this element would be a translating tumbling motion, and there might be very little time to react before the structural element is out of reach of the robot and lost in space. A human operator would have little chance to estimate the object motion,

plan the actions required to bring the robot gripper to the right gripping spot and orientation along the moving element, and complete these actions. However, with the proper equipment, the operator could immediately switch to a mode in which the robot is on its own for recovering the tumbling element in the short time available. To be able to handle such situations without human intervention, the robot could be equipped with a video camera, and could have a database describing the geometry of all the types of structural elements being assembled, with the various goal points which could be reached for a proper grip. While in teleoperating mode, the robot could keep track of which structural element is being handled, so that in case of an emergency the robot would already have retrieved all the relevant information from its database. Analyzing a sequence of images, the robot could find the trajectory of the element in location and orientation. Extrapolating the trajectory to the immediate future, it would plan its own motion to bring its gripper along the trajectory of a goal point of the element. Along this goal trajectory the moving element appears fixed with respect to the gripper, so that the gripping action can be accomplished as if in a static environment.

We have set up an experimental facility which has the necessary components for testing various vision-based control algorithms for intercepting moving objects. These algorithms could be incorporated into the navigation system of a vehicle able to intercept other vehicles, or in a robotic system able to recover objects which are tumbling freely in space . This facility is described in the following section.

2 Experimental set-up

A large *American Cimflex* robot arm, RAMBO, is equipped with a CCD camera and a laser pointer (Figure 1, top left). Images from the camera are digitized and sent to the Connection Machine for processing. A smaller robot arm (*Mitsubishi RM-501*) translates and rotates an object (called the *target* in this paper) through space. Several light-sensitive diodes with focusing optics are mounted on the surface of the object. RAMBO's goal is to hit a sequence of diodes on the moving object with its laser beam for given durations, possibly subject to overall time constraints. Electronics inside the object signal success by turning on an indicator light. Simultaneously, we are developing a full computer simulation in which the camera inputs are replaced by synthetic images (Figure 1, top right).

3 Summary of Operations

The vision-based control loop for RAMBO is shown in Figure 1. We briefly describe the functions of the different modules of this system from data collection to robot motion control, and refer to the sections of this paper which give more details.

1. The digitizer of the video camera mounted on the robot arm can grab video frames when new visual information is needed. A database contains a list of positions of feature points on the target, in the local coordinate system of the target.

2. A low-level vision module extracts locations of feature points from the digitized image (Section 4).

3. An intermediate vision module finds the location/orientation of the target in the camera coordinate system (Section 5, Appendix A and Appendix B).

4. Since the past camera trajectory is known, the position of the camera in the robot base coordinate system when the frame was grabbed is known. The location/orientation of the target is transformed to the robot base coordinate system (Section 6).

5. This most recent target pose at a specific time is added to the list of target poses at previous times. In the Target Motion Predictor, a target trajectory in location/orientation space is fitted to these past target poses and extrapolated to the future to form a predicted target trajectory. We also obtain the predicted trajectories of *goal points* around the target. A goal point is a location –which is fixed in the frame of reference of the target, thus moving in the frame of the robot base– that one of the joints of the robot has to follow for the accomplishment of one of the subtasks of the total action (Section 6).

6. From the predicted goal point trajectories, the Robot Motion Planner calculates the robot motions necessary for following the goal points, and the resulting camera trajectories (Sections 7, 8, 9). If the subtasks are not ordered, the Motion Planner finds an optimal order (Section 10). The camera trajectories are used for transforming subsequent target pose estimates from a camera coordinate system to an absolute coordinate system (Section 6).

4 Low-level Vision

The model-based pose estimation described in the next section requires that feature points be extracted from each of the images of the target. This is the task of the low-level vision module. Feature points in an image could be images of small holes in the target structure, corners of letters, vertices, etc, ... Conversely, the geometric description of the target should contain the 3D locations of the feature points which are easily detected in images.

In our experiments, the target is a polyhedra, and the feature points that we use are the vertices of the polyhedra, so that our image analysis is partly specific to this type of feature detection. Our image processing algorithms involve a sequence of basic local operations [17, 18] implemented on the Connection Machine –enhancement [19], edge detection, edge thinning and vertex detection– followed by some simple processing to determine which pairs of vertices are connected by edges in the image. This last operation proceeds as follows:

Given k vertices, we use the processing cells in the upper-triangle of a $k \times k$ array of cells in the Connection Machine, each assigned to a possible edge between vertices.

1. Enable a grid of $k \times k$ cells. Disable cells in the lower triangle of the array.

2. Copy the addresses of corner points and the incident angles of their edges to the cells in the diagonal of the grid

3. By horizontal grid-scan and vertical grid-scan, spread the incident angles and the addresses of the vertices from the diagonal cells along rows and columns to all the cells in the array.

4. Each non-diagonal active cells (i, j) now has all the information about the i-th and j-th vertices; these cells can determine whether they have a pair of collinear incident edges. If they do, then that vertex pair is marked as being connected (Note that we could also count the number of edge pixels along the line joining the vertices, but this would be much more costly on the Connection Machine and not worthwhile for our purposes).

From this algorithm we obtain a list of all the vertices with for each vertex a sublist of the vertices connected to it. We then produce a list of all the image triples consisting of one vertex with two vertices connected to it. This list of image triples is input to the intermediate-level vision module described in

the next section. The other input is a similar list for the triples of world vertices of the target, from the geometric database describing the target.

5 Intermediate-level Vision: Pose Estimation of the Target

The pose estimation algorithm combines three ideas:

1. Pose estimation by matching triples of image features to triples of target features [20].
2. Standard camera rotations [21].
3. Paraperspective approximation to perspective projection [22, 23].

This combination allows the extensive use of 2D look-up tables to replace the costly numerical computations used by similar previous methods [20, 24–26]. The algorithm is implemented on the Connection Machine.

The feature points detected in an image are grouped into triples (the *image triangles*). Each image triangle can be described by one of its vertices (the *reference vertex*), the length of the two adjacent sides, and the angle between them (the *reference angle*). These adjacent sides do not necessarily have to correspond to actual edges in the image, and all distinct triples of points could be considered, with each triple of points producing three such image triangles. However, if the feature points are vertices, it is useful to only consider image triangles in which the adjacent edges of the reference vertex are actual edges, and to only match these image triangles to world triangles with similar characteristics. This increases the proportion of good matches over the total number of possible matches.

The main algorithm steps are as follows:

1. Each image triangle is transformed by a *standard rotation*. The standard rotation corresponds to the camera rotation around the center of projection which brings the reference vertex of the triangle to the image center. For each reference vertex in the image, rotation parameters are read in a 2D lookup table.
2. Image triangles are then rotated in the image plane around the reference vertex (located at the image center) to bring one edge into coincidence with the image x-axis.
3. Once in this position, an image triangle can be described by three parameters only, the reference angle, the edge ratio (ratio of the lengths of the two edges adjacent to the reference vertex), and a size factor.
4. For each image triangle/target triangle pair, a 2D lookup table can be used to determine the orientation of the target triangle in space. There is one 2D lookup table per target triangle, which gives two possible orientations of this target triangle when the reference angle and edge ratio of its image are entered. Details on the calculations of these tables, based on a paraperspective approximation, are given in Appendix A.
5. Comparing the size of the image triangle to the size of the target triangle of known orientation, we can then find the distance of the target triangle from the camera lens center.
6. The preliminary transformations of the image triangle can be reversed to obtain the actual 3D pose of the target triangle, the corresponding 3D position of the target center, and the image of the target center.

7. The target center projections are clustered to identify the pose of the whole target.

When RAMBO analyzes its first image, it does not have any *a priori* knowledge of which feature triangles are visible. In this case, the system uses all the possible combinations of target triangles and image triangles. However, clustering gives better results if most improper matches are removed, and it is possible to do so after a few consistent pose estimates of the target have been obtained. The system can also avoid considering matches for target triangles which are at a nearly grazing angle with the lines of sight, since for these triangles image analysis is likely to perform poorly and paraperspective does not approximate true perspective well.

Details on the implementation of this pose calculation method on the Connection Machine are given in Appendix B. For a target producing less than 16K image triangle/target triangle combinations, each pose calculation takes around one second on a CM-2 with 16K processors but without floating point processors.

6 Motion Prediction

The computation of a target position from an image gives the translation vector and rotation matrix of the target coordinate system in the camera coordinate system. However, the camera itself is set in motion by the robot arm. The trajectory of the camera in an absolute coordinate system is known, and it is straightforward to get the position of the camera coordinate system at the time the image was taken and to find the target position at this time in an absolute coordinate system.

From a sequence of target positions, the robot must be able to predict future positions of the target in order to construct plans of actions. These target positions are points in six-dimensional space (three translation parameters and three rotation angles), each with a time label. We can fit a parametric function of time, such as a polynomial, to each of these sequences of coordinates. The target trajectory is then described parametrically by six functions of time. Calculating these functions for a future value t of the time parameter will give a predicted target position at this future time.

If RAMBO is used in space and the axes of the frame of reference of the target coincide with the principal axes of inertia, then in the absence of external forces the translation of this coordinate system should be uniform, as well as the rotations around the three axes. But more complex cases could occur (for example, a structural element could be tethered at one end). The data base describing the target could specify what ranges and types of motions are possible, and this data could be used to determine the best way to parameterize the target trajectory.

7 Task and Trajectory Planning

In order to perform task and trajectory planning, RAMBO currently makes the simplifying assumption that a complex goal can be decomposed into a sequence of simple subgoals, and that each subgoal can be performed with one joint of the robot in a fixed position with respect to the target. This joint has to "tag along" with the target, thus we call this joint the *tagging joint* of the robot. The fixed position with respect to the target that the tagging joint must follow to complete a subgoal will be called a *goal point*. All goal points required for each complex action on a target can be predefined in a data base of actions specific to each target. Each goal point is defined by six coordinates, three for the location and three for the orientation of the tagging joint, in the coordinate system of the target.

Once the tagging joint is moving along the target so that it does not move with respect to the target, the more distal joints can be used to perform the finer details required by the subgoal. The programming

of these distal joints will not be considered here, since it is equivalent to programming a robot to perform a task on a fixed object. For example a subgoal for a robot arm on a space shuttle might be grabbing a handle on a tumbling satellite. A database containing the geometry of the satellite would also specify in what position — fixed in the satellite frame of reference — the wrist of the robot arm should be in order for the end effector to grab the handle. Here the tagging joint is the wrist, and the goal point is a position above the handle given in the satellite frame of reference. Once the wrist is positioned at the goal point — which requires constant motion control of the robot arm during the subgoal completion, since the satellite is tumbling — the joints of the end effector require the same grabbing motion with respect to the wrist as would be needed if the satellite were not moving.

In our experimental setup, we have concentrated on reaching the goal points. Each subgoal consists of illuminating a light-sensitive diode mounted on the surface of the target for a given duration. The source of light is a laser pointer mounted on the tool plate of the robot arm. Each diode is mounted inside a tube at the focal point of a lens which closes that tube, so that the laser beam must be roughly aligned with the optical axis of the lens to trigger the electronic circuits which control the output of the diodes. Thus a goal point for the laser tool is defined by the positions at a short distance from the lens of a diode along the optical axis, and by the orientation of this axis.

8 Bringing a Tagging Joint to a Goal Point of the Target

Suppose the original trajectory of the tagging joint in location/orientation space is the vector $\vec{p}_0(t)$ (Figure 2). The goal trajectory in location/direction space is given by the vector $\vec{p}_g(t)$. At time t_0 we want the tagging joint to "launch" from its original trajectory $\vec{p}_0(t)$, and to "land" at time $t_g = t_0 + T$, on the goal trajectory $\vec{p}_g(t)$. The reaching trajectory $\vec{p}_r(t)$ should be equal to trajectory $\vec{p}_0(t)$ at time t_0 and to trajectory $\vec{p}_g(t)$ at time t_g. The operation will last for the reaching duration T. Furthermore, the first derivatives should also be equal at these times, so that the velocities change smoothly when the robot departs from its original trajectory and reaches the goal trajectory. Once a launching time t_0 and a reaching duration T are chosen, the end points of the reaching trajectory $\vec{p}_r(t)$, as well as the first derivatives of the reaching trajectory at these points are known. These boundary conditions are enough to define $\vec{p}_r(t)$ in terms of a parametric cubic spline, a curve in which all the coefficients of the six cubic polynomials of time can be calculated.

In our experiments we have also explored an alternative method which uses a scalar piecewise quadratic interpolation function $f_T(t)$ which is 0 at time t_0, 1 at time t_g, with horizontal derivatives at these times, and continuous derivatives in the time interval. This function is expressed by

$$f_T(t) = 2\left(\frac{t}{T}\right)^2, \qquad 0 \le t \le T/2$$

$$f_T(t) = -2\left(\frac{t-T}{T}\right)^2 + 1, \quad T/2 < t \le T$$

and the reaching trajectory is the interpolated trajectory given by

$$\vec{p}_r(t) = f_T(t-t_0)\vec{p}_g(t) + (1 - f_T(t-t_0))\vec{p}_0(t)$$

Note that the predicted motion of the target and the predicted goal point trajectories should be updated every time a new target pose is found for the target. After each of these updates, the reaching trajectory of a tagging joint should be recomputed based on the new goal trajectory, and the present joint trajectory, which may itself be a reaching trajectory started after a previous update.

9 Optimizing Reaching Trajectories

With either method of estimating reaching trajectories, one difficult problem is the preliminary choice of T, the duration of the reaching trajectory. Duration T should be chosen so that the resulting linear and angular velocities and accelerations are within the limits imposed by the robot design. Also, the reaching trajectory should not cross an obstacle or the target itself, and should not require the robot to take impossible configurations. Usually, time is the rarest commodity, and the shortest time T compatible with the above constraints should be chosen.

In our present simulations on a serial machine, the duration T is simply calculated as the time which would be necessary if the reaching trajectory followed a linear path, at a constant velocity chosen to be a safe fraction of the maximum linear velocity of the robot. Finally, we check whether this trajectory crosses robot limits or causes a collision with the target. If it does, the reaching trajectory is recalculated with a safe intermediary goal instead of the final goal point.

A better optimization of the reaching trajectory would require a choice of duration T which would set the velocities and accelerations along the reaching trajectory close to the limit capabilities of the robot. This can be done by calculating the reaching trajectories for a series of durations T, finding the maximum of the second derivatives along the trajectories, and identifying the trajectory with the smallest duration T which does not require positions, velocities and accelerations beyond the robot capabilities. On the Connection Machine, this operation can be done in only a few steps. We set up a 2D array of processing cells with time as the vertical dimension. Every column of the array contains a copy of the predicted goal trajectory, with the first cell containing the position of the goal at the present time in location/direction space, the next cell the position at a time increment in the future, and so on. Every column also contains a copy of the trajectory of the tagging joint, sampled with the same time increments as the goal trajectory. The difference between columns is that they use different durations T of the reaching trajectory, increasing from one column to the next.

Each cell computes a point of the reaching trajectory for the time t corresponding to its row and for duration T corresponding to its column, and then computes estimates of appropriate derivatives at its reaching trajectory point by communicating with its neighbors in the column. The maxima of the derivatives are computed for each column. The column that has the smallest duration T and for which the maxima of the positions and derivatives do not violate robot limits is the column which contains the desired reaching trajectory. The near-term future motion of the robot should be controlled based on this selected trajectory.

10 Higher Level Planning

In a complex action we have a set of tasks A, B, C, D, each of which requires a tagging joint to move smoothly to a specific goal trajectory. In some actions the order of the tasks is not specified, and we have to choose a good order in which to carry out the tasks. "Good" order here means one which minimizes the total time spent moving between goal trajectories. Notice that this is not equivalent to a travelling salesman problem, since the time required to move from performing, say, task B to task D, depends on when we move from B to D, therefore depends on the sequence of tasks which have been performed before B.

10.1 Greedy Approach

At any given time, we can compute all the reaching trajectories to all the goal trajectories of the remaining n tasks. From among these n reaching trajectories, we can choose the one with the shortest duration, and pursue the corresponding task. This could possibly be repeated in real time after each task is accomplished, but does not guarantee the best overall sequence of tasks.

However, given that our model of the anticipated motion of the target is being updated as new information arrives, this procedure may make the most sense, in that there may be no point in computing a global optimal sequence of tasks based on a model of anticipated target motion which will not remain correct in the future.

10.2 Exhaustive Search and Dynamic Programming

Assuming, however, that our model of anticipated motion is accurate enough to allow a meaningful computation of an overall optimal sequence of tasks, one (unattractive) possibility is to compute the total time required to complete all the tasks for all possible orderings of the tasks. For n tasks this will involve computing

$$\sum_{i=1}^{n} \frac{n!}{(i-1)!}$$

best reaching trajectories.

A dynamic programming method has been developed which can precompute the best overall sequence of n tasks using computation proportional to

$$\sum_{i=1}^{n} \frac{n!}{(n-i)!\,(i-1)!}$$

For four tasks, for example, this approach would require computing 32 reaching trajectories rather than the 64 required for the exhaustive approach.

11 Conclusions

We have described research on robots acting in dynamic environments. We discussed the use of vision to assess the motion of objects relevant to the robot's goals, and the use of particular motion prediction and planning techniques to accomplish these goals. We described a set of experiments currently under way involving a robot arm equipped with a camera and laser operating on a single moving target object equipped with light sensors. Finally, we detailed the parallel implementation of many of the tasks involved in the accomplishment of goals in dynamic environments, including parallel image processing, parallel pose estimation, and parallel planning.

12 Acknowledgements

The support of the Defense Advanced Research Projects Agency and the U.S. Army Engineer Topographic Laboratories under Contract DACA-76-88-C-0008 is gratefully acknowledged.

REFERENCES

[1] M. Asada, Y. Fukui, and S. Tsuji, "Representing a Global Map for a Mobile Robot with Relational Local Maps from Sensory Data", International Conference on Pattern Recognition, 1988, 520-524.

[2] R.A. Brooks, "Solving the Find-Path Problem by a Good Representation of Free-Space", *IEEE Trans. Systems, Man, and Cybernetics*,13, no.3, March-April 1983, 190-197.

[3] J.L. Crowley, "Navigation for an Intelligent Mobile Robot", *International Journal of Robotics Research* ,1, March 1985, 31-41.

[4] S.J. Dickinson and L.S. Davis, "An Expert Vision System for Autonomous Land Vehicle Road Following", Computer Vision and Pattern Recognition, 1988, 826-831.

[5] B. Faverjon and P. Tournassoud, " A Local-Based Approach for Path Planning of Manipulators with a High Number of Degrees of Freedom", Proc. IEEE Robotics and Automation, 1987, 1152-1159.

[6] M. Kabuka and A.E. Arenas, "Position Verification of a Mobile Robot using Standard Pattern", IEEE Journal of Robotics and Automation, 3, 1987, 505-516.

[7] T. Lozano-Perez, "A Simple Motion-Planning Algorithm for General Robot Manipulators", *IEEE Journal of Robotics and Automation*, 3, June 1987, no.3, 224-238.

[8] H.P. Moravec, "Sensor Fusion in Certainty Grids for Mobile Robots", *AI Magazine*, 9 (2), Summer 1988, 23-104.

[9] K. Sugihara, "Some Location Properties for Robot Navigation using a Single Camera", *Computer Vision, Graphics and Image Processing*, 42, 1988, 112-129.

[10] C. Thorpe, M.H. Hebert, T. Kanade, and S.A. Shafer, "Vision and Navigation for the Carnegie-Mellon Navlab", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10, 1988, 362-373.

[11] M.A. Turk, D.G. Morgenthaler, K.D. Gremban, and M. Marra, "VITS–A Vision System for Autonomous Land Vehicle Navigation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10, 1988, 342-361.

[12] K. Fujimura and H. Samet, "A Hierarchical Strategy for Path Planning among Moving Obstacles", Center for Automation Research CAR-TR-237, University of Maryland, College Park, November 1986.

[13] K. Fujimura and H. Samet, "Accessibility: A New Approach to Path Planning among Moving Obstacles", Computer Vision and Pattern Recognition, 1988, 803-807.

[14] Q. Zhu, "Structural Pyramids for Representing and Locating Moving Obstacles in Visual Guidance of Navigation", Computer Vision and Pattern Recognition, 1988, 832-837.

[15] N. Kehtanrnavaz and S. Li, "A Collision-Free navigation Scheme in the Presence of Moving Obstacles", Computer Vision and Pattern Recognition, 1988, 808-813.

[16] P.S. Schenker, R.L. French, D.B. Smith, "NASA telerobot testbed development and core technology demonstration", Space Station Automation (IV), SPIE vol.1006, Cambridge, MA, November 1988.

[17] D.H. Ballard and C.M. Brown, *Computer Vision*, Prentice-Hall, 1982.

[18] A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, second edition, Academic Press, New-York, 1982.

[19] D. Harwood, M. Subbarao, H. Haklahti, H. and L.S. Davis, "A New Class of Edge Preserving Filters", *Pattern Recognition Letters*, 6, 1987, 155-162.

[20] S. Linnainmaa, D. Harwood, D. and L.S. Davis, "Pose Determination of a Three-Dimensional Object using Triangle Pairs", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10, 1988, 634-647.

[21] K. Kanatani, "Constraints on Length and Angle", *Computer Vision, Graphics and Image Processing*, 41, 1988, 28-42.

[22] Y. Ohta, K. Maenobu and T. Sakai, "Obtaining Surface Orientation of Plane from Texels under Perspective Projection", Proc. IJCAI, 1981, 746-751.

[23] J. Aloimonos and M. Swain, "Paraperspective Projection: Between Orthography and Perspective", Center for Automation Research CAR-TR-320, University of Maryland, College Park, May 1987.

[24] D.W. Thompson and J.L. Mundy, "Three-Dimensional Model Matching from an Unconstrained Viewpoint", Proc. IEEE Robotics and Automation, 1987, 208-220.

[25] D.W. Thompson and J.L. Mundy, "Model-Directed Object Recognition on the Connection Machine", Proc. DARPA Image Understanding Workshop, 1987, 98-106.

[26] R. Horaud, "New Methods for Matching 3-D Objects with Single Perspective Views", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9, no.3, May 1987.

APPENDIX A
Lookup tables for the pose calculation of a triangle from an image

A triangle is defined by two sides from a common vertex (the reference vertex) and the angle between them (reference angle). We assume that a standard camera rotation and a camera roll have brought the reference vertex of the image triangle to the image center and one side parallel to the X-axis of the image. Notations for the known angles and side lengths of the image and world triangle are shown in Figure A1. Three (yet unknown) numbers define the position of the world triangle in space:

1. Angles θ_1 and θ_2 of the lines P_0P_1 and P_0P_2 with respect to the Z-axis.
2. Distance R from the center of projection O to the vertex P_0 on the Z-axis. Once the angles have been obtained the calculation of R is straightforward and does not require a table.

The perspective transformation which yields the image triangle from the world triangle is approximated by a paraperspective transformation, which amounts to a sequence of two transformations, a local orthographic projection of the world triangle on a plane parallel to the image through its reference vertex, and a perspective transformation from the resulting image to the actual image (Figure A2).

Looking at Figure A2, we see that

$$\frac{f}{R} = \frac{d_1}{D_1 \sin\theta_1} \tag{1}$$

Similarly, for the image segment p_0p_2 and P_0P_2

$$\frac{f}{R} = \frac{d_2}{D_2 \sin\theta_2} \tag{2}$$

We then define the parameter s, the ratio of the two sides of the image triangle, the parameter S, the ratio of the two corresponding sides of the space triangle, and the parameter K, the ratio of s and S :

$$s = \frac{d_1}{d_2},\ S = \frac{D_1}{D_2},\ K = \frac{s}{S}$$

The parameter K is a known constant of the problem, since the dimensions of the image triangle and of the world triangle are both known. Dividing Equations 1 and 2, we find that the ratio of the sines of the angles θ_1 and θ_2 must be equal to this constant K:

$$K = \frac{\sin\theta_1}{\sin\theta_2} \tag{3}$$

The dot product of the two unit vectors $\vec{n}_1$ and $\vec{n}_1$ parallel to P_0P_1 and P_0P_2 is equal to $\cos\alpha$. These two unit vectors have components $(\sin\theta_1, 0, \cos\theta_1)$ and $(\sin\theta_2\cos\phi, \sin\theta_2\sin\phi, \cos\theta_2)$, where ϕ is the angle of p_0p_2 with the X-axis. The dot product is

$$\cos\alpha = \sin\theta_1\sin\theta_2\cos\phi + \cos\theta_1\cos\theta_2 \tag{4}$$

The angles θ_1 and θ_2 are unknown. The fact that the ratios of the sines must be equal to a known constant (Equation 3) allows the elimination of one of the unknowns. The result is a second degree equation in $\sin^2\theta_1$. We define $X_1 = \sin^2\theta_1$ and find

$$\sin^2\phi\, X_1^2 - (K^2 - 2K\cos\alpha\cos\phi + 1)X_1 + K^2\sin^2\alpha = 0 \tag{5}$$

Equation 5 always has two positive solutions, but only the smaller solution is smaller than 1 and can be made equal to a sine. Thus we always get a single solution for $\sin\theta_1$

$$\sin\theta_1 = \left[\frac{-B - \sqrt{\Delta}}{2\sin^2\phi}\right]^{1/2} \tag{6}$$

with

$$B = -(K^2 - 2K\cos\alpha\cos\phi + 1)$$

$$\Delta = B^2 - 4K^2\sin^2\alpha\sin^2\phi$$

This single sine solution gives two solutions θ_1 and $\pi - \theta_1$, which correspond to mirror image directions of P_0P_1 with respect to the plane parallel to the image plane through P_0. The corresponding value for $\sin\theta_2$ found from Equation 3 also gives two solutions θ_2 and $\pi - \theta_2$. Note that we cannot combine the two θ_1 solutions and the two θ_2 solutions in all four ways. Indeed each of the two θ_1 solutions corresponds to cosines of opposite signs (similarly for θ_2), so that two combinations of θ_1 and θ_2 give a positive number for product $\cos\theta_1\cos\theta_2$, whereas the other two give a negative product. To choose which two combinations out of the four are the correct ones, we rewrite Equation 4

$$\cos\theta_1\cos\theta_2 = \cos\alpha - \sin\theta_1\sin\theta_2\cos\phi$$

and we check the sign of the right-hand side. Finally, the distance R of the vertex P_0 from the center of projection is computed using Equations 1 or 2. For each target triangle, a table is built with two entries, one for the image angles ϕ, the other for the ratios K between the image and world triangle side ratios. The values read can be the (θ_1, θ_2) solutions, or expressions which require the values of θ_1 and θ_2. If there are 50 characteristic triangles in a target object, 50 tables are precalculated. Each table would be of size 120×128 if image angles ϕ are quantized in 1.5 degree increments from 0 to 180 degrees, and if 128 values of K are considered. Since variations of θ_1 and θ_2 are small for K large, a log scale is used for K. The distance R, which completes the description of the pose of the triangle in space is obtained from θ_1 or θ_2, and from the ratio of a target edge length to its image edge length (Equation 1 or 2).

Appendix B
Pose calculation on the Connection Machine

To run the pose estimation algorithm described in Appendix A, the Connection Machine is used in three ways:

1. as a lookup table engine
2. as a combinatorial machine, for taking care of all the combinations of target triangles and image triangles in parallel
3. as an image processor, for calculating convolutions and finding peaks in 2D Hough transform space.

1. Lookup table engine:

We set up 2D arrays of processing cells. For the standard rotation computation, each cell represents a pixel of the image, and contains the rotation parameters required when the reference vertex of an image triangle is located at that pixel, and also contains the rotation parameters for the inverse rotation.

For the computation of the target triangle orientations, a different 2D array is precomputed for each target triangle. The vertical dimension corresponds to different reference angles of the image triangles, in equal increments; the horizontal dimension coresponds to different edge ratios of the image triangles, in log scale because variations of target triangle orientation are small when the edge ratio is large. Since we are only interested in clustering the target centers, each cell skips the computation step of finding the target triangle orientation and directly determines the coordinates of the vector joining the target triangle reference vertex to the target center.

2. Combinatorial machine:

The lookup tables previously described have been created once and for all for a given target geometry and are then used repeatedly as new images of the target are analyzed. We now describe the distribution of data and operations at run time. We again set up a 2D array of processing cells. Rows are assigned to target triangle data. The coordinates of the reference vertex in the target coordinate system and the edge

ratio of each target triangle are copied to all the cells of a specific row prior to run time. Columns are assigned to image triangle data, which are refreshed after each image capture and analysis. The coordinates of the three vertices of an image triangle are copied to all the cells of a specific column. Also copied are the standard rotation and inverse rotation parameters fetched from the standard rotation lookup table. Consequently, each cell within the range of columns and rows filled up by target and image triangles contains a different combination of target triangle and image triangle. Each cell can then:

- transform the image triangle by standard rotation and swing around the optical axis, finding the new edge ratio and reference angle
- find the vector from the reference vertex to the target center from a lookup table
- find the distance of the target triangle vertex
- use previous information and the inverse rotations to find the coordinates of the target center and its image, and the rotation matrix of the target.

3. Image processor:

We set a 2D array in which each image pixel is assigned to a cell. The bin-count of a cell is incremented if the projection of the target center falls on the corresponding pixel.

The image is then smoothed. Each cell having a bin-count above a given threshold is considered a candidate cluster. These cells are numbered by decreasing bin-count. This number is broadcast back to the 2D array of image triangles and target triangle combinations which contributed to the corresponding clusters. The other cells of this array are disabled.

The 2D clustering of the images of the target center might not yield the correct target pose. Thus we also cluster the target center with respect to its Z-coordinate. We set up a 2D array with one row of cells per selected image cluster and one column per increment of the Z-coordinate. A bin-count increment for a cell of this array occurs when a cell of the image and target triangle array contains the cluster number corresponding to the row and the z-coordinate corresponding to the column. Then within each row we perform a 1D smoothing. The row which contains the highest cluster is selected, and in the image and target triangle array only the cells which contributed to this cluster are selected.

Finally, among these selected cells, the average value of the position of the target center is found, while dropping the outliers. The average rotation matrix of the target center is also calculated at this point. Our experiments with a simple polyhedra gave unambiguous results without further clustering in about one second, but experiments with more complex objects might require additional clustering with respect to the rotation matrix.