REAL-TIME, MODEL-BASED TRACKING OF THREE-DIMENSIONAL OBJECTS

Gilbert Verghese
Charles R. Dyer

Computer Sciences Technical Report #806

November 1988

Real-Time, Model-Based Tracking of Three-Dimensional Objects

Gilbert Verghese[1]

Charles R. Dyer

Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Abstract

In this paper we present new algorithms for real-time, model-based tracking of three-dimensional (3D) objects which are represented using polyhedral models. Given a previous viewpoint solution, this is the problem of continuously revising the camera viewpoint parameters based on the steadily changing image of a 3D object. We do not consider the problems of initially identifying the object or calculating the initial viewpoint; we only consider the dynamic aspects of 3D tracking. We assume a single object is moving arbitrarily in space, permitting changes in all 6 degrees of freedom. The object can be moving at an arbitrary velocity, but we assume a dense sequence of images so that the object has both temporal and spatial continuity. We do not assume any knowledge of the motion parameters, however.

Two algorithms are described in detail. Each algorithm is implemented as a parallel program running on two tightly-coupled multiprocessors — a Sequent Symmetry for high-level processing (HLP) and an Aspex Pipe for low-level processing (LLP). The two algorithms differ in that one uses a camera parameter hypothesize-and-test approach and the other uses dynamic edge tracking for camera parameter adjustment. The hypothesize-and-test algorithm uses the HLP to hypothesize projections from slightly different viewpoints and the LLP cross-correlates these hypotheses with successive images. For each set of hypotheses, the HLP computes a revised set of viewpoint parameters based on the best cross-correlation results. In the second algorithm the current projected-view of the model is sent to the LLP by the HLP; the HLP incrementally readjusts the viewpoint based on results of continual local shifting of the projected points with respect to the edge points detected in the images. This dynamic edge tracking is performed by the LLP. Predictive methods which assume bounded 3D translational and angular accelerations are used whenever speeds exceed the bounds determined by the camera parameter update rate.

---

[1] Current address: Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

The support of the National Science Foundation under Grant Nos. DCR-8520870 and IRI-8802436, and the National Aeronautics and Space Administration under Grant No. NAGW-975 is gratefully acknowledged. Programming assistance by Karey Gale was invaluable to the completion of this work. Code for the Newton-Raphson convergence procedure was provided by David Lowe.

1. Introduction

Given a dense sequence of images of a known three-dimensional object which is moving arbitrarily, we describe algorithms for real-time tracking. The use of a polyhedral model of the object we are tracking distinguishes this problem from other work which is not model-based. That is, other methods for deriving structure from motion, structure from image flow, and structure from multiple views, for example, do not assume knowledge of the object in view. Hence mainly low-level features such as points, edges and blobs have been used as the basis for motion analysis. Even in model-based approaches, authors have usually assumed that their models contain a set of distinguishable local features such as vertices [Asad84] and blobs [Gilb80; Orou80].

The motivation for our approach is the following scenario: If the image sequence is dense with respect to the speed of object motion, once objects are recognized the task becomes one of continuously tracking the changing parameter values needed to keep the projection of the 3D object model in correspondence with the dynamic images. In this steady-state situation recognition of object structure is not a problem since the model is already known; only the motion parameters are unknown. (This is in contrast with other work such as [Boll87] which assumes known motion but unknown objects.)

We make the assumption that the temporal sampling rate is high with respect to the rate of spatial changes in the images. This means we can take advantage of the frame-to-frame coherence between a projection of the model in one image and its slightly differing projection in successive image frames. As a result, processing can take advantage of both top-down, goal-directed knowledge of the given 3D object model, as well as bottom-up, data-driven feature tracking based on spatio-temporal continuity of the visible object.

In this project we use a polyhedral model of the object to be tracked. We start with a given model and a known correspondence between the camera coordinate system and the object

coordinate system. Motion is described as changes in the camera parameters. Arbitrary motion is allowed, so there are six degrees of freedom in position and orientation to be updated over time. This approach is most similar to the work of Gennery [Genn82; Genn86]. He did not use the dense sampling assumption, however, so his algorithm was based on also computing velocity and angular velocity estimates in order to predict a single hypothesized projection of the model at each cycle. He also assumed that there were no abrupt velocity changes. Our tracking algorithms have camera parameter adjustment rates which are slightly less than the frame rate. When object points' position-changes in between camera updates are known to be at most a small value, there is no need for velocity predictability assumptions. We exhaustively search a neighborhood of the point's previous position. Thus the generality of our approach relies on maximizing the camera parameter update rate for a given of size of the search neighborhood.

There is a tradeoff between the camera update rate and the size of the search neighborhood for each point. Assuming random object velocity, the number of operations required to compute feature correspondences for camera update is no less than δ^2, where δ is the maximum object feature displacement in the image plane between successive camera parameter updates. This distance is the product of the time, t, elapsed since the previous solution and the maximum velocity, v, of object features in the image plane (i.e. $v = \delta/t$ and $t \geq \delta^2$). Since sub-pixel displacements are not detectable, $\delta \geq 1$. v is thus a maximum at $\delta = 1$. Thus we choose to implement algorithms which consider maximum object feature displacements of no more than one pixel in the image plane.

If object velocity is known *a priori*, prediction can be used to adjust the viewpoint parameters in near constant time. This theoretically removes the upper bound on v by making t independent of δ. However, since prediction error is proportional to δ and errors must not be allowed to propagate, predictive algorithms require error correction phases which are more time-consuming than direct, single-pixel displacement algorithms. Hence predictive methods assume stable velocities in order to allow larger object speeds, while single-pixel displacement methods

restrict maximum object speeds in order to allow random motion. Our algorithms are hybrid single-pixel displacement methods which use prediction in those cases where object feature speed exceeds the $1/t$ limit in the image plane.

In order to minimize t and the use of error-prone prediction techniques, we have developed efficient parallel algorithms which are implemented on two tightly-coupled multiprocessors, an 8-stage Aspex Pipe and a Sequent Symmetry. Processes running on the two machines are synchronized to coordinate the various processing cycles.

In addition to Gennery's work, there are a few other methods which have been developed for solving similar problems. Asada *et al.* [Asad84] tracked moving polyhedral objects in 3D blocks-world scenes. Their work assumed perfect line drawings as input and each line drawing image was labeled independently before the correspondence between images was performed based on a "junction transition table" to define legal correspondences. O'Rourke and Badler [Orou80] used a 3D model of the human body to predict and track image sequences of human motion. Their basic feedback loop between high and low levels using an hypothesize-and-test paradigm is very similar to one of our approaches. In contrast, however, at each cycle their system generated a single best prediction of the 3D structure of the human model and then generated a set of image regions-of-interest in which to search for predicted features. They did not use dense sampling and they assumed that only a few locally-distinguishable features (e.g. hand and foot) were to be tracked from frame to frame.

The rest of the paper is organized as follows. Section 2 presents an overview of the tracker system organization including a description of the multiprocessor system organization. Sections 3 and 4 describe two different tracking algorithms, their implementations and their complexity analysis.

2. Tracking System Organization

In this section we briefly describe the system architecture which was used for implementing the tracking algorithms. This system consists of two multiprocessor subsystems, an 8-stage Aspex Pipe and a Sequent Symmetry as shown in Figure 1. This two-level organization corresponds to the two kinds of processing requirements in computer vision. The low-level, iconic image processing is performed by Pipe and the high-level symbolic processing is done using the Sequent.

Pipe is a linearly-connected set of eight pipeline processing stages [Kent85]. Each stage contains local image buffer memory and specialized functional units for performing point (lookup table), neighborhood (convolution), and arithmetic and Boolean operations. All operations are performed on 256 by 256 images. The computations in the stages are all performed in one frame-time (16.67 msec) so that communication between stages is synchronous. Each stage is directly connected to its two nearest neighbors. In addition, there are six busses permitting non-nearest neighbor communication.

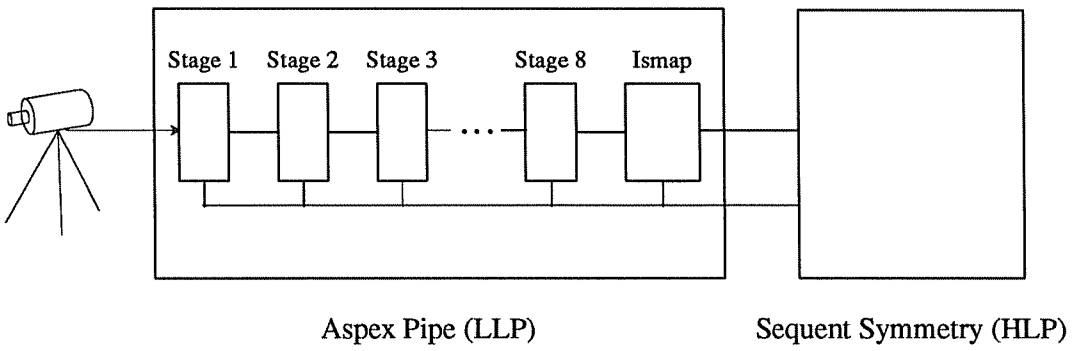

Figure 1. System architecture used for real-time tracking.

It is important to note that Pipe is not a conventional pipeline or systolic architecture because each stage has image buffers which can store images for an arbitrary period of time between computations. Thus Pipe may be used not only for conventional pipeline processing but also for iterative and top-down processing.

Video and Input stages on Pipe allow direct digitization and input of images at video rates. Output from Pipe to the Sequent Symmetry is done through the Ismap processing stage. In addition to transmitting images to and from the Sequent, Ismap can also compute histograms, cumulative histograms and inverted histograms of an image. Ismap's image and histogram buffers are memory-mapped into the Sequent's address space so that these buffers can be directly read/written by either Pipe or the Sequent.

The Sequent Symmetry is a tightly-coupled, shared-memory multiprocessor. Although 20 processors are available, in this project we used only a single processor for all high-level processing. In this paper we will refer to Pipe as the low-level processor (LLP) and the Sequent as the high-level processor (HLP).

Both algorithms are implemented as tightly-coupled parallel programs on the LLP and the HLP. Each algorithm has the following feedback loop control structure. The LLP detects edges in a newly input image, producing an edge map. The LLP then combines this edge map with an image produced by the HLP by computing the view of the model using the current camera parameters. This combination of iconic model and edge data is converted into a symbolic form using the Ismap processor of the LLP. From this symbolic form, the HLP computes measurements required for updating the camera parameters. The next cycle begins with a new projection being sent from the HLP to the LLP. Communication between the two processors is asynchronous.

3. Tracking using Camera Parameter Hypothesize-and-Test

In this section we describe a 3D model-based tracking algorithm which uses the HLP to hypothesize new camera parameters. Figure 2 shows the feedback organization of the major parts of this algorithm as well as the relationship to the hardware configuration. The camera parameters are used to generate a set of projected models which are cross-correlated with edges detected in the latest image. The results of cross-correlation by the LLP are then used by the HLP to select the best match and update the camera parameters. The algorithm also uses the cross-correlation results to decide when it has lost track of the object. In this case it predicts the object's position and orientation (pose) based on the assumption that the object has moved with bounded acceleration. Thus we attempt to also track objects moving at speeds exceeding the limits determined by a given update rate. Prediction is also useful when occlusion prevents tracking for a short period of time.

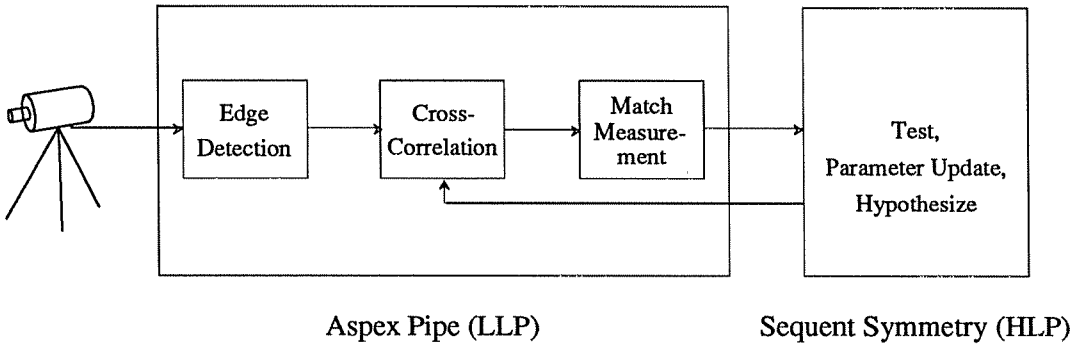

Figure 2. Hypothesize-and-test tracking algorithm.

3.1. Algorithm Description

Our approach assumes that a recently correct viewpoint is given and that any bounded, six degree-of-freedom motion has occurred. In order to adjust the viewpoint, we consider twelve hypotheses which treat each of the six camera parameters independently. We found that with dense sampling and dense hypothesis-verification, exact motion solutions are not necessary. Simultaneous motion in several dimensions is resolved by alternate stepwise motion in each dimension. Hence tracking can be done without considering all possible combinations of motions and their magnitudes. For each dimension, we determine whether or not a small positive or negative change has occurred. That is, we generate a pair of new views of the model by incrementing and decrementing a single camera parameter. These two model hypotheses are each then cross-correlated with edge points extracted from the current frame. If there is a significant difference in these results and the largest correlation indicates a match, the camera parameter is updated. If motion in a particular degree of freedom is persistent, i.e., maintains the same direction for several iterations, the cross-correlation results can also be used to determine when the speed limits imposed by the update rate have been exceeded. This information is used to choose between "direct" and "predictive" modes of operation. This is described in detail below.

The algorithm cycles through hypothesize-and-test phases in four degrees of freedom, three in rotation and one in scale, each phase taking 4 frame-time units. The two degrees of freedom in translation parallel to the image plane are considered differently because of the hardware available in the LLP. In particular, planar translation is performed every 4 frame-time units in parallel with each of the four phases above.

The two hypothesize-and-test phases, one for rotation and scale, and one for planar translation, are discussed in the next two subsections. Each involves hypothesis generation, cross-correlation with the current edge map, and updating the given camera parameter as their

major computational components.

3.1.1. Rotation and Scale Changes

Hypotheses are considered cyclically for changes in the camera z-coordinate, rotation about the x-axis, rotation about the y-axis, and rotation about the z-axis. In order to approximate camera parameter changes which cause single-pixel motion of projected model-line-segment endpoints in the image for a given degree of freedom, we calculate the following. Positive and negative changes to the camera z parameter are proportional to the current camera z parameter due to similar triangles. Reasonable values for positive and negative changes to angles of rotation are also proportional to z since $\sin(\theta) \approx \theta$ for small θ. In each cycle, we first compute the image obtained by interpolating between endpoints of the model line segments viewed from the current camera pose, appropriately clipped at image boundaries. Two hypotheses are produced by projection after an increment and a decrement of the current parameter by the value which causes single-pixel motion of projected endpoints.

Edges are detected using a thresholded, magnitude-of-gradient operator. The resulting edge points are then "expanded" to occupy a 3 by 3 neighborhood of pixels. The expansion is required in order for the results of cross-correlation to differentiate between no motion and a loss of correspondence between the model and the object. Since the two hypotheses associated with a given degree of freedom are at most distance 2 apart, if all pairs of hypotheses are highly correlated, the object must be stationary. Low correlation between pairs of hypotheses indicates the object and viewpoint are not in correspondence. Without expanding edge points, both cases would result in low correlation and disambiguation would require an additional hypothesis of "no motion." This gain in efficiency is at the expense of parameter precision.

In order to determine the degree of match between the edge map for the current image and an hypothesis map, these two binary images are cross-correlated. A global measure of correlation is then computed by summing the 1's in the cross-correlation and normalizing using the total

number of model points. Since hypotheses are considered in pairs, two match measures are computed for each cycle.

Next, we use two different approaches to updating the camera parameters for rotation and scale. These correspond to the two cases associated with direct and predictive hypothesis generation. In the direct case, each hypothesis' match measure is compared to a model-specific threshold to decide whether to accept or reject this hypothesis. The pair of matches for each degree of freedom is then evaluated to determine how to update the associated camera parameter. Namely, if either match in a pair is accepted and one match measure is significantly higher than the other, the parameter is updated to correspond to the viewpoint for the best match. If both matches are rejected but the camera parameter has been recently repeatedly updated, it is assumed that the speed in the associated degree of freedom has exceeded the limit imposed by the update rate. This is reasonable since a sudden match failure cannot result from a decrease in speed. Prediction which assumes bounded acceleration is attempted in this case. If neither of these two cases holds, the parameter is left unchanged.

Above we stated the conditions when dense tracking fails but recent tracking history indicates motion may have exceeded acceptable speed bounds. In this case we can predict motion using the persistence assumption and motion information from previous frames. Again, this is performed independently for each of the four degrees of freedom in rotation and scale. Smoothly changing velocity is permitted. We predict the future value of the changing camera parameter based on the known sampling rate and an initial velocity estimate. We then hypothesize two new parameter values which are derived from the predicted value. These are generated in the same way as described above for the direct case. The cross-correlations and match measures are computed and tested in the same manner as given above. Here they are used to determine how to update the associated camera parameter velocity estimate. If either match measure is significantly higher than the other, the velocity estimate is incremented or decremented accordingly. Otherwise, the velocity estimate is left unchanged. If the velocity

estimate returns to a value that is within the limits allowed by the update rate, direct hypothesis generation, testing, and parameter updating is resumed. However, if both matches are repeatedly rejected, this signals failure of prediction beyond recovery by our system. At this point, the tracker halts because the most recent successful match is too outdated.

3.1.2. Planar Translation Changes

Hypotheses for model shifts parallel to the image plane are generated by shifting the current projected model using the LLP. Since the computation of these hypotheses is very efficient, these two camera parameters are updated at four times the rate of the other four parameter updates (i.e., every 4 frames instead of every 16 frames).

Four hypotheses are generated corresponding to one pixel shifts in the horizontal and vertical directions. Due to the projective geometry these shifts are independent of scale and simply correspond to increments and decrements in the camera x and y parameters. Each hypothesis is cross-correlated with the edge map computed during the current cycle, and a match measure is produced. The normalized and thresholded global match measure for each hypothesis is computed in the same way as described above for the other degrees of freedom.

Updating the camera parameters for x-axis and y-axis shift is performed similarly to the case-based method for updating the other camera parameters. Tight coupling enables the HLP to access the four match measures for hypotheses made by the LLP and update both camera parameters in one cycle.

When prediction is required, the HLP maintains parameter velocity estimates for x and y. The current projected model which is shifted by the LLP to generate the hypotheses takes predicted planar translation into account. Hence the hypotheses can be generated, tested and used to update velocity estimates in the same manner as rotation and scale.

3.2. Implementation, Analysis and Results

The previous section specified the computational components and their interactions during execution of this model-based tracking algorithm. In this section we focus on the synchronization and flow of data between the Sequent (HLP) and Pipe (LLP). This will also lead to a derivation of the camera parameter update rate for this algorithm.

Consider a single cycle of the algorithm in which hypothesize-and-test phases in one degree of freedom in rotation or scale and both degrees of freedom in planar translation are performed. The parallel operation of the processors begins with the LLP performing edge detection on the input image sequence and the HLP projecting the two hypothesized model views. The LLP produces a new edge map every two frame times. Having written the hypotheses into a buffer on the LLP, the HLP signals that hypotheses are ready and waits for the corresponding match measures. At the frame time following this signal, the LLP cross-correlates the hypotheses with the most recent edge map and computes the respective match measures using Ismap's histogrammer. At the next frame time the LLP informs the HLP that the two match measures are ready. The LLP also shifts the projection in the four planar directions, cross-correlates these with the edge map, and computes the four planar translation match measures in Ismap. Meanwhile, the HLP has read the two previous match measures and is waiting for the LLP to finish computing the planar translational match measures. This is signaled by the LLP when it starts the next cycle. At the beginning of the next cycle the HLP reads the match measures for planar translation hypotheses, updates the camera parameters, computes the new hypotheses, and reprojects the model.

The time required to complete one cycle defines the parameter update rate for the algorithm. Since the LLP uses two frame times to compute the edge map, one frame time to compute the match measures for rotation or scale, and one frame time to compute the match measures for planar translation, each cycle takes 4 frame times to complete. Three camera parameters are

updated every 4 frame times (or one every 22.23 msec on average). However, each planar translational parameter is updated every 4 frame times (66.68 msec), and each rotation and scale parameter is updated every 16 frame times (266.72 msec).

Since the imposed bounds on the object's speed depend on the object's distance from the camera, the camera's focal length, and the size of each pixel, we describe the tracking ability of our algorithm in terms of the speed of the object's projection on the image plane. Our algorithm tracks object motion of at most 15 pixels/sec in planar translation and 3.75 pixels/sec in rotation and scale. When speeds exceed these bounds, the respective acceleration bounds for successful prediction are 15 pixels/sec^2 and 3.75 pixels/sec^2, assuming no abrupt direction changes.

Figure 3 shows four images of results of a tracking sequence using the algorithm. Each image contains a set of slightly differing hypotheses overlaid on the most recent input frame. Since these hypotheses were projected using the current camera parameters, each image shows the correspondence between the model and the actual polyhedral object at several successive instances during the total tracking interval.

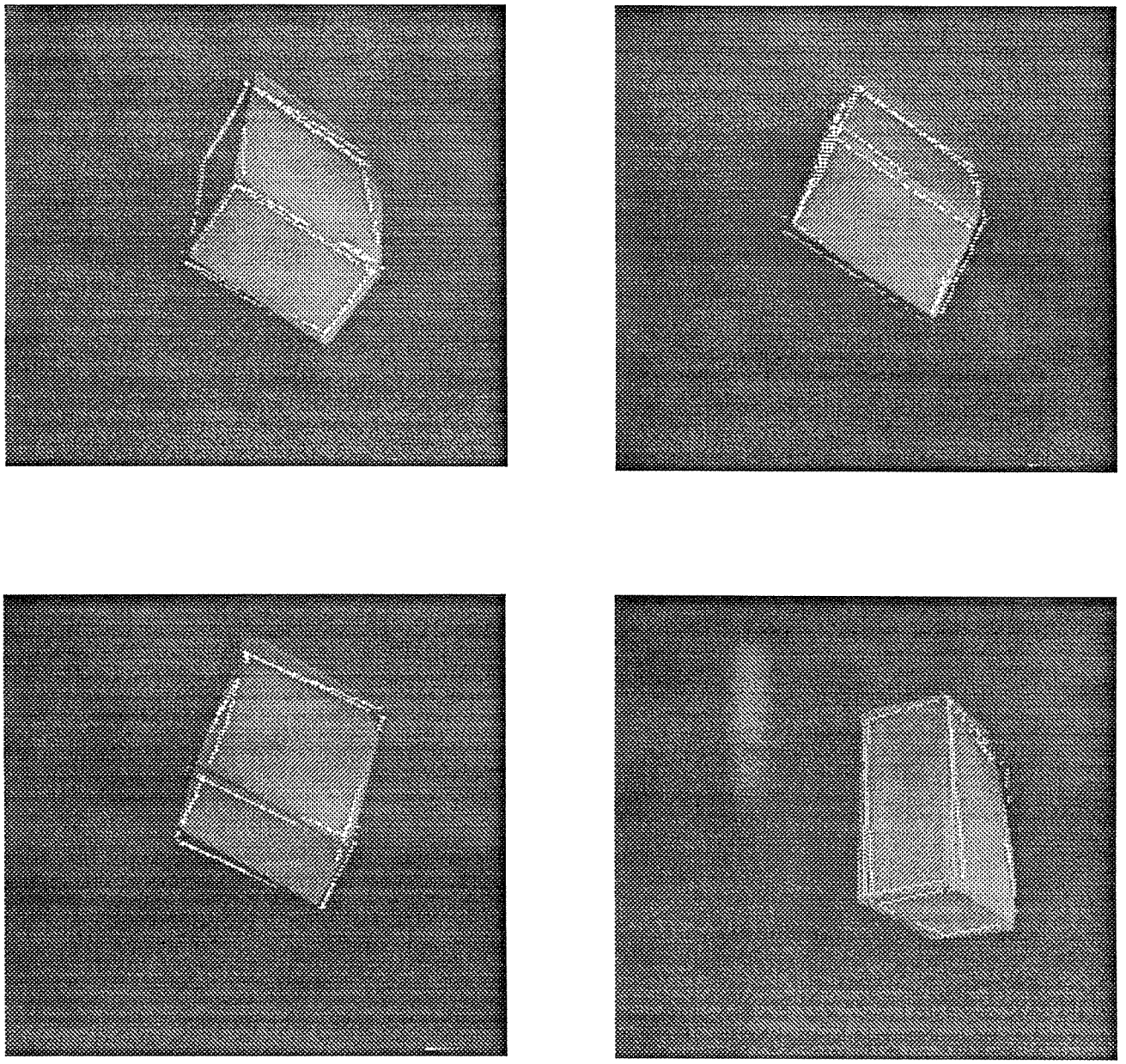

Figure 3. Selected frames during a real-time tracking sequence using the hypothesize-and-test algorithm. Several successive model hypotheses are shown overlaid on each frame.

4. Camera Parameter Adjustment using Dynamic Edge Tracking

In this section we describe a second approach to the 3D model-based tracking problem. The algorithm divides the problem into dynamic edge tracking and model line-segment tracking subproblems. Figure 4 shows the overall feedback organization of the algorithm and its configuration on the LLP and HLP. In each cycle of this algorithm, the current projection of the visible edges of the model is sent to the LLP from the HLP. The HLP incrementally readjusts the camera parameters based on results of continuous local shifting of the projected points with respect to the edge points detected in the frame sequence. This dynamic edge tracking is performed by the LLP at half the frame rate. Edges are detected at this rate also using a thresholded magnitude-of-gradient operator.

Model-based segment tracking is performed on the HLP using an adaptation of Lowe's viewpoint solution algorithm [Lowe85]. Since Lowe's method assumes only a single image is given and the goal is recognition, it must search for a candidate pairing between data edges and model segments before attempting to solve for viewpoint. Having a recent solution we avoid this

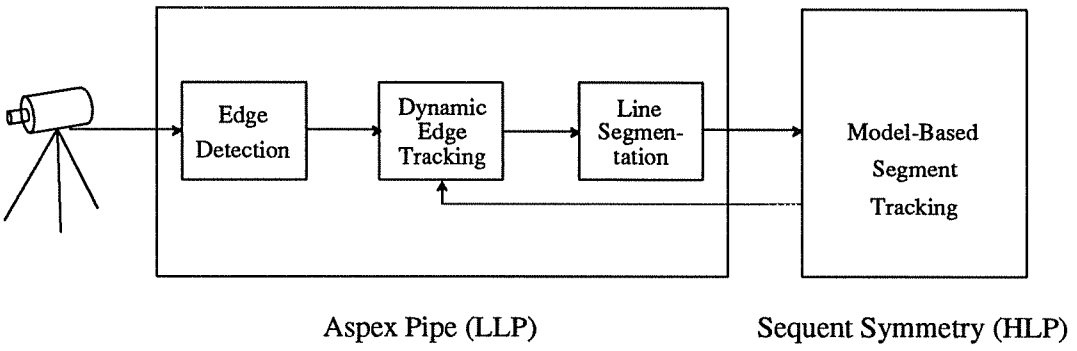

Figure 4. Camera parameter adjustment using dynamic edge tracking.

search in our adaptation. Also, rather than converging to a fixed solution, our algorithm allows a "moving target." That is, between camera updates during the iteration, it is assumed that the object may have moved and that the dynamic edge tracker has since updated the positions of object edge segments. Furthermore, the algorithm detects when object speed exceeds the limit imposed by the dynamic edge tracker and then initiates prediction.

4.1. Dynamic Edge Tracking

The 2D motion correspondence problem is to find the displacement between points projected onto the image plane at different times by the same physical point. For non-points, it is well known that only the perpendicular component of motion correspondence can be determined using local information. The dynamic edge tracking problem is the restriction of the general problem to perpendicular components. This is useful for model-based tracking by the following argument. In determining segment correspondence, it is not important to determine the parallel component of all correspondences since this can be computed from the endpoints if required. However, exact endpoints are not required for model matching since the perpendicular components are often sufficient to constrain the motion of polyhedral objects containing many segments at many orientations. Hence the dynamic edge tracker can be restricted to perpendicular components.

This problem is easily solved for consecutive binary edge maps in a dense frame sequence: For each edge point in the first map we search in a neighborhood centered on the corresponding point in the second edge map. We describe a solution to the dynamic edge tracking problem which is also useful for model-based segment tracking. In order to do so, we first define a *segment map* to be an image containing a view of the visible edge segments in a model, where each projected model segment is labeled uniquely. All other points of the segment map are unlabeled. Given a binary edge map E and a segment map S, we compute a new segment map having a labeled point if there is a nonzero point in E within one pixel of a labeled point in S,

and all labeled points in a 3 by 3 neighborhood of the current point in S have the same label. In this case, the resultant value in the new segment map is the uniquely determined label. Figuratively, each segment "sticks to" nearby image edge points and "shrinks" away from other segments.

The dynamic edge tracker repeats this computation using the new segment map and a new edge map derived from the next image frame. After nk frame times, where k is the time required to compute a new segment map, a segment may have moved n pixels in any direction as it "follows" its coincident edge points. Consequently, a complete solution to this problem must also verify that the resulting segment map is actually a real view of the model. That is, the data-driven segment updating on the LLP is useful only if the results are verified with the model on the HLP. For example, noise or background edge points which are connected in the edge map to object edges are not distinguishable by a local process; as a result, such points may be incorrectly labeled as segment points. As another example, consider the case of object segment disocclusion; the dynamic edge tracker incorrectly labels points on the newly visible segment as belonging to a different, previously occluding edge.

The HLP performs the required verification using the model. This is described in more detail in the following section. If it becomes necessary, the HLP may also replace the current segment map in the LLP with a correct view of the model (as was done for each hypothesis in the hypothesize-and-test algorithm).

4.2. Model-Based Segment Tracking

In this section we describe a segment matching algorithm for the HLP using line segment information contained in a 3D polyhedral model and a segment map. Rather than using complete line segments, our approach is based on comparing the endpoints of line segments only. Endpoint information is extracted from segment maps by inverted histogramming in the LLP using the Ismap processor. Since each data segment has a unique label, the inverted histogram

collects the coordinates of all pixels having each label into an ordered list. This list can then be efficiently searched by the HLP for data segment endpoints. Note that the label identifies the corresponding model segment. For each data segment, the verification procedure uses the corresponding model segment's projection to determine whether the extracted endpoints are plausible. In order to account for occlusion of parts of these segments, the endpoints are projected onto the model segment's 2D projection, and a measure of the perpendicular distance between the segments is computed. This is normalized and compared to a threshold value to determine whether the data segment is acceptable. If so, it is used later to help constrain the viewpoint solution process.

If there are enough acceptable segments, the HLP then continues to solve for viewpoint. On the other hand, degradation of the segment map as described in the previous subsection may leave too few acceptable data segments. In this case, the HLP replaces the dynamic edge tracker's segment map with a projection of the model in which each segment is uniquely labeled. Since the dynamic edge tracker operates continuously, this is done by projecting the model into a buffer and informing the dynamic edge tracker of its availability. At the appropriate time, the LLP then replaces the segment map.

Given the dynamic edge tracker's segment map, the HLP interprets the segments' endpoint information to obtain pairings between model segments and labeled segments in the segment map. Using the current camera parameters and this set of pairings, it then determines if some projection of the polyhedral model brings the paired segments into spatial correspondence. In doing so, it computes adjustment values for all camera parameters. Note that this procedure is tolerant to noise since the match need not be perfect. Furthermore, the spatial correspondence measure does not depend on endpoint proximity, only on parallelism and perpendicular distance between the segments. Hence the set of pairings may be sparse and may include partially-occluded edges.

The correspondence method is based on the Newton-Raphson iterative technique [Lowe85]. Each iteration attempts to bring image edge points (x, y) into correspondence with projected model segment points. The partial derivatives of x with respect to each of the six camera parameters are evaluated. The error measure is defined as the sum of each partial derivative times the respective unknown parameter error correction value. Similarly, an error measure is also defined for y. Doing this for three distinct image edge points and setting all error measures to 0 gives a complete linear system. Gaussian elimination with row pivoting is used to solve for all six camera parameter adjustments. Least squares techniques are used if more than three points are available. After this system is solved, the camera parameters are updated. Each iteration results in roughly an order of magnitude reduction in the parameter error. However, there is no advantage in iterating to a solution using the same (outdated) data segment points at each iteration. Instead, the algorithm extracts new data segments from the dynamic edge tracker for each iteration. Thus, successive camera parameter adjustment values also indicate velocity in the six degrees of freedom. This is important for the purpose of motion prediction, which is discussed next.

Since the camera adjustment values computed at each iteration can be used to estimate persistent motion in each degree of freedom, they are useful in assessing the need for prediction. When the HLP cannot extract enough useful segment points from the LLP's segment map but there has been persistent motion, it is assumed that object speed exceeds the limit imposed by dense sampling. A sudden dynamic edge tracking failure cannot result from a decrease in speed. In this case, prediction is carried out in a manner similar to that used in the hypothesize-and-test algorithm. Velocity estimates in each degree of freedom are maintained. They are initially determined by an average of recent adjustment values. Camera parameter estimates computed from these velocities are used to predict the new camera parameters. The model is then projected using these predicted parameters to provide the LLP with a new segment map, which it updates using the most current edge map. The HLP then extracts line segments from this segment map,

performs an iteration of Newton-Raphson convergence, and updates the velocity estimates based on the direction and magnitude of resulting adjustments. This continues until the velocity estimates diminish to values which are within the limits of the dynamic edge tracker. If, on the other hand, the dynamic edge tracker continually produces segment maps from which too few endpoints can be extracted, the algorithm halts because the most recent successful solution is too outdated.

4.3. Implementation, Analysis and Results

The implementation of this algorithm involves slightly different synchronization and data flow compared to the hypothesize-and-test algorithm. The model-based segment tracker is implemented on the Sequent (HLP) and the dynamic edge tracker is implemented on Pipe (LLP). Both operate independently, with most of the data flow in the bottom-up direction. Flow in the opposite direction is not predetermined as it was for the hypothesize-and-test algorithm. It occurs only when the model-based segment tracker determines that the segment map must be replaced. This is implemented by having the dynamic edge tracker test at the beginning of each cycle whether there is a new segment map available to replace the previous one. The HLP computes the new segment map and signals the LLP when it is ready.

A cycle begins with the HLP producing a segment map by projecting the model using the current camera parameters. The LLP uses this segment map and a current edge map to compute a new segment map. This is done in two frame times. Each segment map is also sent to Ismap which computes an inverted histogram of the segment map. Hence Ismap operates in parallel, also at half the frame rate. The HLP polls Ismap to determine when the inverted histogram is ready. At that time it searches the sorted lists of data segment points for endpoints of all model segments projected. If enough endpoints are found, the HLP does one iteration of Newton-Raphson convergence, updates the camera parameters, and polls Ismap for a new inverted histogram. Otherwise, it reprojects the model creating a new segment map, waits for the LLP to

process this segment map and produce a subsequent one, and only then polls Ismap for a new inverted histogram.

Since the Newton-Raphson technique can determine large camera adjustments relative to those accumulated over a short period of time, the time spent polling does not degrade tracking performance in the non-predictive mode. Tracking performance is determined by the update rate of the dynamic edge tracker and not by the camera parameter update rate in this case. The dynamic edge tracker updates its segment map once every 2 frame times (33.34 msec). Since the imposed bounds on the object's velocity depend on the object's distance from the camera, the camera's focal length and the size of each pixel, we describe the tracking ability of our algorithm in terms of the speed of the object's projection on the image plane. Our algorithm guarantees successful tracking of arbitrary object motion up to 30 pixels/sec.

When predictions are made, one iteration of Newton-Raphson convergence is performed between updates. This takes less than 4 frame times on the Sequent. Hence when speeds exceed 30 pixels/sec, the algorithm can track an object whose acceleration is 15 pixels/sec^2 in any degree of freedom, assuming no abrupt direction changes.

Figure 5 shows four images at various points during a tracking sequence. The current segment map is shown overlaid on the most recent input frame, showing the correspondence between the current model and the actual polyhedral object.

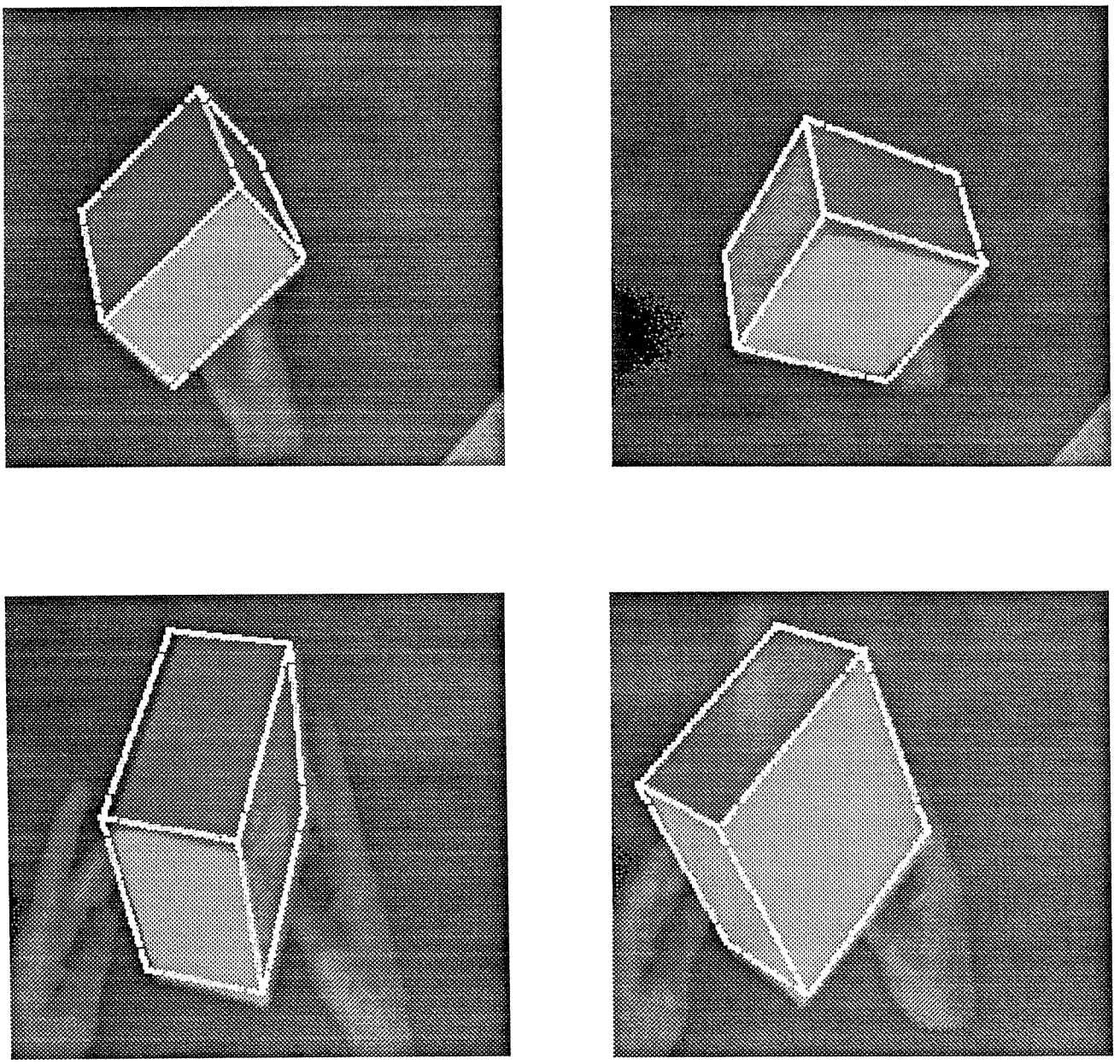

Figure 5. Several frames of a tracking sequence with overlays showing model segments derived from the current camera parameters.

5. Concluding Remarks

We have presented two algorithms which advantage of the spatio-temporal continuity of objects for motion tracking by using temporally-dense sampling and simple model parameter update procedures. This was made possible by a tight feedback loop between specialized high-level and low-level vision processes. Two kinds of feedback loops were represented by the two algorithms. In the hypothesize-and-test loop, high-level model projections were compared directly to low-level edge maps. This approach has a high tolerance for noise and selects only pertinent image features for matching. In the camera parameter adjustment and dynamic edge tracking feedback loop, segments are used as an intermediate-level representation for comparing model and image features. The continuity of the data segments over short periods of time is used for data-driven tracking. Segment endpoints are used to adjust the model parameters at selected times.

Both algorithms exhaustively search a small neighborhood of each feature to be tracked and avoid any assumptions about expected motion. This is in contrast with most previous model-based approaches which rely on computing a single best prediction.

Areas for further investigation include parallelizing the high-level processes on the multiprocessor HLP and extending the system to track multiple moving objects.

References

[Asad84] M. Asada, M. Yachida and S. Tsuji, Analysis of three-dimensional motions in blocks world, *Pattern Recognition* **17**, 1984, 57-71.

[Boll87] R. Bolles, H. Baker and D. Marimont, Epipolar-plane image analysis: An approach to determining structure from motion, *Int. J. Computer Vision* **1**, 1987, 7-55.

[Genn82] D. Gennery, Tracking known three-dimensional objects, *Proc. National Conf. on Artificial Intelligence*, 1982, 13-17.

[Genn86] D. Gennery, Stereo vision for the acquisition and tracking of moving three-dimensional objects, in *Techniques for 3-D Machine Perception*, A. Rosenfeld, ed., North-Holland, New York, 1986.

[Gilb80] A. Gilbert, M. Giles, G. Flachs, R. Rogers and Y. U, A real-time video tracking system, *IEEE Trans. Pattern Analysis and Machine Intelligence* **2**, 1980, 47-56.

[Kent85] E. Kent, M. Shneier and R. Lumia, Pipe — Pipelined image processing engine, *J. Parallel Distrib. Comput.* **2**, 1985, 50-78.

[Lowe85] D. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer, Boston, 1985.

[Orou80] J. O'Rourke and N. Badler, Model-based image analysis of human motion using constraint propagation, *IEEE Trans. Pattern Analysis and Machine Intelligence* **2**, 1980, 522-536.

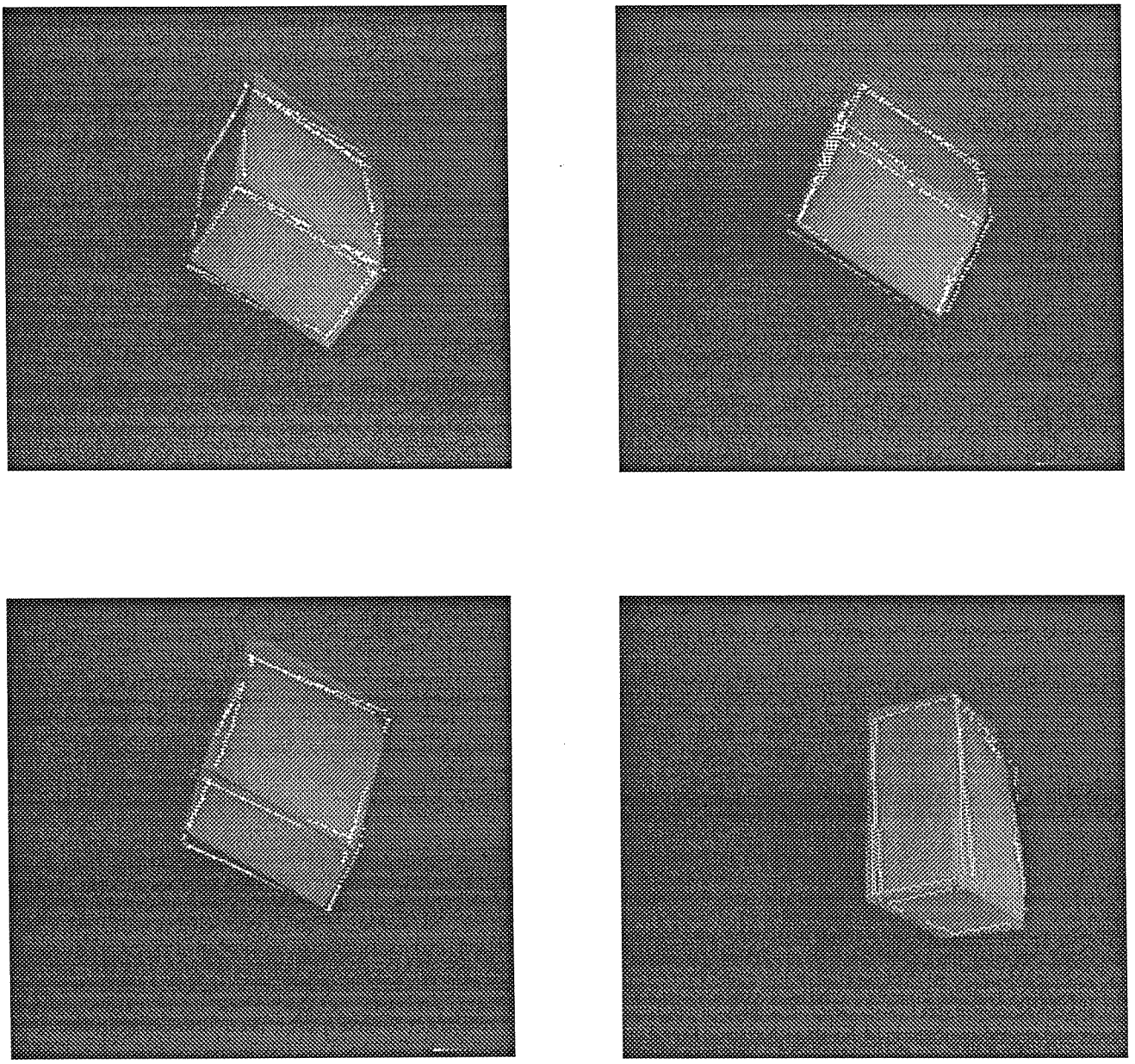

Figure 3. Selected frames during a real-time tracking sequence using the hypothesize-and-test algorithm. Several successive model hypotheses are shown overlaid on each frame.

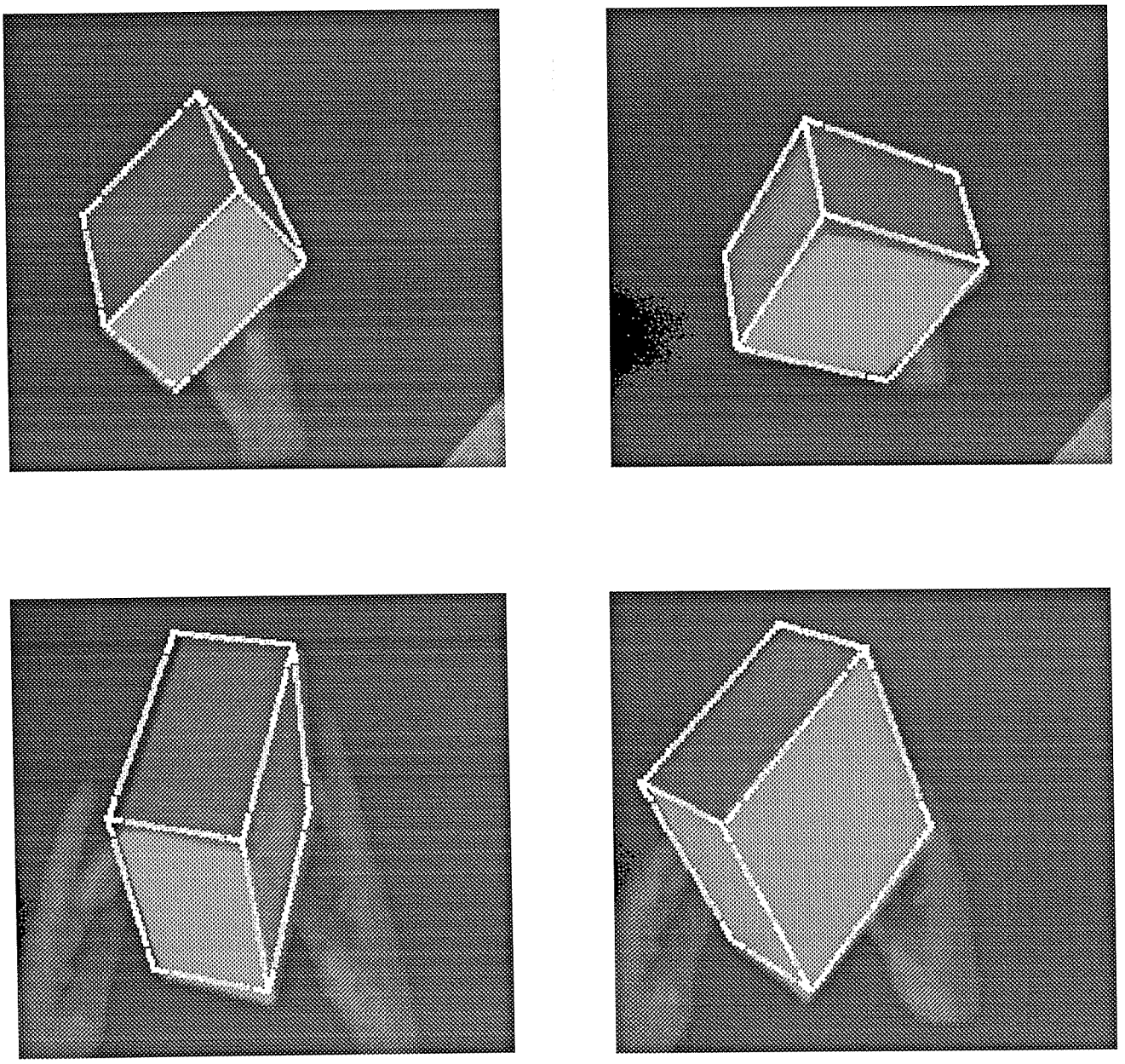

Figure 5. Several frames of a tracking sequence with overlays showing model segments derived from the current camera parameters.