Real-time Model-based Tracking Using Perspective Alignment

Gilbert Verghese John K. Tsotsos
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A5
email: verghese@vis.toronto.edu

Abstract

Perspective Alignment is a new method for performing *back-projection*, the key update step in monocular three-dimensional (3D) model-based tracking. This tracking problem requires the 3D position and orientation (pose) of a known rigid object to be maintained using features found in a stream of two-dimensional (2D) images. The motions of 2D image features are tracked by local, high-speed processes, and their displacements are used by the Perspective Alignment back-projection algorithm to compute the modeled object's changing 3D pose in real-time. Our real-time implementations of the feature-tracking sub-system, the Perspective Alignment sub-system, and their tightly coupled interaction are described.

Improvements over conventional methods are achieved by taking advantage of Perspective Alignment's ability to provide constrained, bounded estimates of model pose in impoverished imaging situations. Thus, we need not rely on predictability of 3D motion in many of these situations. Results of on-line, real world object tracking in impoverished, unpredictable imaging situations are shown, and the system's performance limits are given.

1. Introduction

A brute-force approach to the monocular 3D rigid model-based tracking problem is to re-recognize the object for each consecutive 2D image frame. However, since the solution to the combinatorial problem of pairing model features with image features is not likely to change abruptly, we can usually solve the tracking problem without combinatorial searches. Many tasks require continuous 3D pose representation of moving objects, which in turn requires fast special purpose processes for tracking. Hence it can be said that the main goal of tracking is to avoid the computational complexity of re-recognition whenever and however possible. Each correct image-model feature pair greatly reduces the complexity of finding a globally consistent pairing. Therefore a tracking system ought to maintain as many pairings as possible if it cannot solve fully for model pose, and it ought to maximally constrain the model to simplify recovery of lost pairings. Perspective Alignment [Verghese 93] is the first method to achieve this, and this paper presents the first real-time implementation of the method.

If the continuous motion of image features were computable, then the tracking problem would in fact be linear,. The more realistic assumption of discrete image feature displacements leads to the classic non-linear tracking problem [Dickmanns 90]. Several solution methods have been developed over the years, but each one depends on a sufficient number of independent image-model feature pairs to determine 3D rigid-body pose. This includes the work of [Lowe 91], which uses predicted data as a substitute for missing data in order to obtain sufficient constraint for a full pose estimate. The work we describe here relaxes this assumption. Our new Perspective Alignment algorithm is able to partially recover pose when feature constraints are insufficient for a complete solution. One main advantage is that such partial solution may serve the purpose of the task and avoid the immediate need for re-recognition. Another advantage is that it permits eventual recovery of lost image-model feature pairs without the full combinatorics of recognition. Our new 3D model-based tracking method continues progress that we have witnessed in approaches to this problem over the years. This background is summarized below to provide a context for our approach and to put our work in perspective.

2. Previous Work

[Wallace and Mitchell 80] computed 3D motion of modeled objects using 2D Fourier Descriptors extracted from each frame. The Descriptors were normalized and compared to those in a library of Normalized Fourier Descriptors (NFD's). The library covered the orthographic viewpoint sphere in a manner similar to an Aspect Graph model description. Orientation was tracked through successive matching of data NFD's with library NFD's. This method was one of the first to track 3D orientation in real-time. The main drawback was that time-consuming line-segmentation of closed contours and a 1D Fourier transform were required for each frame. Furthermore, the method could not tolerate any occlusion.

[Gennery 82, 86] proposed an algorithm that is the precursor of most of today's locally-linear approximation methods. It generated a single 3D pose based on velocity and angular velocity estimates computed from previous model poses, and then projected the model into 2D. The projected model features directed the image search for the corresponding 2D features. Computed displacements were incorporated into a Kalman filter which updated the velocity and pose parameters, making a linear estimate of the next pose. An elegant back-projection-prediction process was introduced with the Kalman filter. However, image-model feature pairing was lost between sampled frames, the recovery of which entailed a time-consuming image search. Another drawback was that the filter imposed a non-optimal linear trajectory on object motion prediction.

[Thompson and Mundy 88] used a clustering method similar to a Hough Transform. Model pose hypotheses were generated until correspondence with the configuration of image features was verified. This hypothesis-test search took a spiral shaped path through pose parameter space. Testing was done with a computationally intensive clustering or voting method. This relatively brute-force back-projection method limited the performance of the system. More accurate and precise results could be achieved with less demanding iterative back-projection processes. Subsequent approaches did not use clustering.

[Lowe 90, 91] described a system in which articulated objects were modeled and tracked. This system had the same control structure as Gennery's, but several components were improved. Rather than using a Kalman filter, Lowe proposed a diagonal variance matrix to augment the Jacobian, arguing that the filtering property of the resulting system was appropriate for the tracking problem, while the more computationally demanding Kalman filter limited the range of accelerations that could be handled. Unlike Gennery's fixed size search window, the variances bounded the size of the 2D feature searches to confidence intervals that could vary over time. Lowe used predicted data as a substitute for missing data in order to obtain sufficient constraint for a full pose estimate in case of underdetermination. Incorrect predictions would move the model *away* from the solution and hamper the tracker's recovery. Hence this was not a suitable way of dealing with missing data. The main drawback, as with Gennery's system, was that image-model feature pairings were not maintained between back-projection attempts.

[Verghese et al 90A, B] described two tracking systems for polyhedral objects. The first used hypothesize-and-test, and the second used Lowe's locally-linear iterative approximation for back-projection. Dense sampling of the image stream was performed using a real-time PIPE image-processor. Feature tracking occurred on the PIPE at a much higher frequency than back-projection. Thus, unlike previous systems, model feature correspondences with image features were maintained between back-projection attempts, avoiding the 2D feature search in cases of *short-range* motion. Previous *long-range* methods did not produce continuous descriptions of object motion. However, short-range processes had spatiotemporal limits beyond which tracking failed.

Several other methods used the same paradigm as Gennery and Lowe: [Wu et al 89], [Bray 90], [Daucher et al 93]. The methods of [Wu et al 89] and [Dickmanns 90] used dynamic (i.e. kinematic) modeling for trajectory-prediction. [Daucher et al 93] split the back-projection computation into model orientation (attitude) and translation steps. However, multiple attitude solutions complicated the translation step when features were few.

Our work is a next step in the development of tracking systems. Previous systems lose track of objects unnecessarily. They lose track if some global match method (e.g., image correlation, clustering, matrix solution) fails. However, we note that it is possible to keep the model's correspondence, or *alignment*[1], with one or more local features despite the lack of a global match. Furthermore, new alignments can naturally extend existing ones to eventually produce a global solution. Hence the alignment framework we propose is ideal for the tracking problem. We begin with an overview of our real-time alignment tracking algorithm. Then we describe its components and their interactions: feature tracking, feature validation and ordering, Perspective Alignment back-projection, and feature tracker calibration.

[1]The term *alignment* has also been used by [Huttenlocher and Ullman 87, 90] to describe orthographic verification of hypotheses in an interpretation tree search for recognition.

3. Algorithm Overview

We model objects using 3D line-segments, augmented with self-occlusion information for fast hidden-line clipping. We make the standard tracking assumptions: The objects to be tracked may contain curved surfaces, but must contain some straight edges. These edges project to straight image line-segments through standard perspective projection geometry. The model and the object in view are initially calibrated, which establishes an initial image-model feature pairing. The object may then undergo arbitrary rigid motion. The Perspective Alignment algorithm can incorporate constraints from point and corner features as well as line segments, but the other stages of our implementation are currently limited to line segments.

Figure 1 shows the major concurrent processes and their interaction. At the highest level, the algorithm is a pipeline. Each stage in this pipeline transforms the data as it flows through the stage. Each stage operates at a different frequency and maintains its own results for other stages to use. Earlier stages operate at higher frequencies than later stages, therefore care is taken to ensure coherence of data structure transfer between stages.

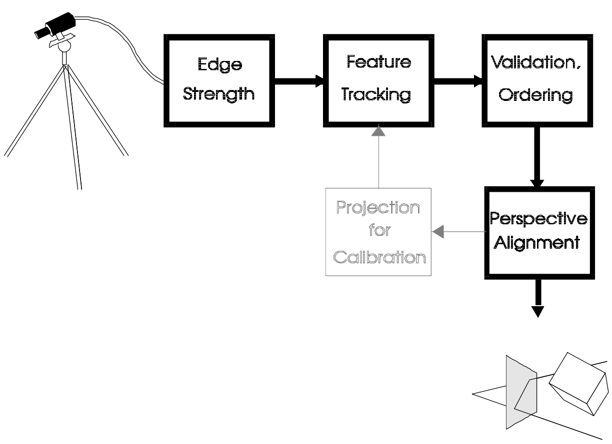


Figure 1. Tracking System Pipeline Components

3.1 Edge Detection Stage

The first stage is edge-detection. It is performed on a Datacube MV20 using the built-in magnitude-of-gradient operator. The data structure it produces is a 30 Hz stream of 512×480 8-bit edge strength images. These images are memory-mapped into the data space of general-purpose host computer programs.

3.2 Feature tracking Stage

The feature tracking and subsequent stages are performed by the host computer (our configuration currently has two hosts, a SUN SPARC II and a Silicon Graphics Power Series 4D/380VGX with 8 processors). The data structure that the second stage maintains is a list of 2D line-segments. The goal of the feature tracking stage is to maintain each segment's correspondence with image edge peaks. The procedure samples the edge strengths along perpendiculars to each line segment for locating peaks that indicate the new position of the image line-segment. It then updates the line-segment's coordinates according to the least-squares linear fit to the peak locations. Peak location allows the method to operate in a variety of lighting conditions by avoiding image intensity or gradient thresholding.

3.3 Validation Stage

The validation stage maintains and monitors the *persistence* and *reliability* of all line-segments produced by the feature tracker in order to sequence the image-model feature pairs for the Perspective Alignment back-projection stage. The output data structure is an ordered sub-list of image line-segments and corresponding paired model edges. A line-segment's *persistence* is measured by the number of frames in which the segment has been reliably tracked by the feature tracker. A line segment's *reliability* is measured by its length divided by the RMS error ($+\in$ to avoid overflow) of its sampled perpendicular edge strength peak positions. A combination of persistence and reliability measures (currently a fixed linear combination that favours persistence) determines line-segment ordering.

After preference ordering is established, line-segment junction constraints from the model are imposed on image segment endpoint coordinates. For each two-segment junction, the intersection point replaces the two endpoints. For three-or-more-segment junctions, the common junction point is perpendicularly projected onto the involved line-segments in order of least preferred to most preferred segment. Hence the most persistent and reliable segment endpoints move the least amount relative to their image edges. This is preferable to choosing the centroid of intersections as the junction of three or more segments. Junction points are not required by the Perspective Alignment back-projection algorithm, but we have found this constraint useful in propagating the reliability of good segments to others.

3.4 Perspective Alignment Stage

The Perspective Alignment back-projection stage performs the transformation from 2D features to 3D object pose parameters. As stated at the outset, this is a nonlinear problem to which several solution methods have been proposed. Our solution produces results essentially equivalent to previous solution methods when a sufficient set of features is provided, and a constrained, bounded pose parameter solution when fewer features are provided. The algorithm is very briefly described below. For a more detailed description, see [Verghese 93].

The Perspective Alignment back-projection stage considers features in the order provided by the previous stage. Analytic relationships between real-world features (in this case line-segments) and their perspective projections are used to allow each feature in the sequence to contribute the maximal additional geometric constraint to the previous constraints on object pose. Thus, as many degrees of freedom as possible are determined given the available feature information.

The algorithm is a decision tree with no backtracking. As we step through the decision tree, we add constraints that further instantiate the model's degrees of freedom. If all the model's degrees of freedom become instantiated at some depth, then the pose of the model is fully determined. If this does not occur before we finish processing a leaf node, then we are left at a partial solution which maximally constrains the model. This is preferable to back-projection failure. Previous methods are incapable of producing such partial solutions.

See Figure 2. An object edge is always coplanar with its perspective projection in the image. This plane can be determined without the object edge. It is the plane defined by the viewpoint or camera origin (pinhole) and the image edge. Therefore, the corresponding model edge should be placed within this constraint plane. In that case, the model edge is *aligned* with its corresponding image edge. This simple constraint is applied repeatedly for subsequent image-model feature pairs. However, a more complex algorithm is required for each additional alignment since previous alignments must not be violated. These algorithms are described in [Verghese 93] and are summarized below. Any number of features can be aligned, with extraneous feature alignments incorporated into the pose solution in case of overdetermination.

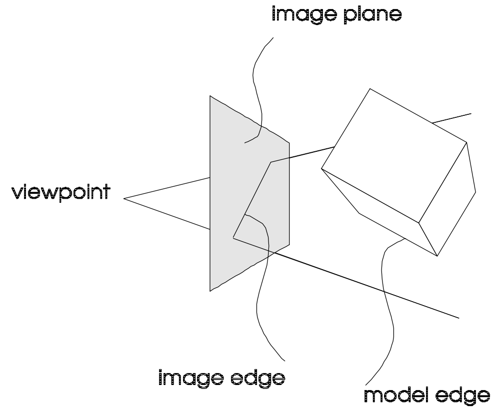


Figure 2. Perspective Alignment LINE Constraint

3.4.1 Aligning Edge 1

The algorithm begins by aligning the model with the first (i.e., most persistent and reliable) edge in the sequence. The model is translated from one endpoint of the first model edge to its plane of constraint via the shortest path (perpendicularly). The model is then rotated by the smallest angle formed by the model edge and its plane of constraint. The rotation axis is determined by the plane normal's cross-product with the model edge's direction vector. Rotations about specific axes are performed with unit quaternions. The translation and rotation described place the first model edge in its plane of constraint. In the general case, four degrees of freedom remain for the model's pose, two translational (in the plane) and two rotational (about the edge itself and about the plane's normal).

3.4.2 Aligning Edge 2

The second most persistent and reliable segment and its paired model edge are considered next. We seek a model pose solution for which the first and second model edges are in their planes of constraint.

Since the first model edge is already in its plane, imagine turning the model about the first plane's normal. This keeps the first edge in its plane but varies the orientation of the second model edge if the edges are not perpendicular. If they are perpendicular, then we can immediately determine a feasible target orientation for the second model edge, namely, the cross product of the second plane's normal and the first edge's direction vector since the cross product of two vectors is perpendicular to both. If the two edges are not perpendicular, then a trivial solution is obtained by turning the model about the first plane's normal until the first model edge is parallel to the second plane. Then there are necessarily two feasible target orientations for the second model edge. They are obtained by computing the intersections of the second plane and the second model edge's direction vector's cone of rotation about the first model edge's direction vector. The intersections exist over two symmetric intervals of edge 1's orientations about the two trivial orientations that make edge 1 parallel to plane 2. The two intersections coincide at the extremities of the intervals, where the plane is tangent to the cone.

In each case and over each contiguous solution interval of edge 1 rotations about plane 1's normal, it is possible to track one or two target orientation solutions for edge 2 about edge 1. We plotted a locus for each track and displayed it in real-time to study its shape. Each locus was smooth with low spatial frequency. Figuratively, the model edges slide within their planes of constraint as we rotate edge 1 about normal 1, solve for edge 2's orientation for each track, and translate in plane 1 to place edge 2 in plane 2. The smoothness of each locus allows us to coarsely sample the ranges without losing information.

If we have run out of features at this point in the decision tree, then several factors are taken into consideration in choosing a pose: the deviations of target from predicted orientations, the deviations of target from predicted model centroids, the *covering* constraint, which is mentioned at the end of subsection 3.4.4, and continuity of prediction.

The prediction method is a simple linear model centroid trajectory for translation and a linear quaternion rotation about a stiff axis through the centroid. This expresses a more natural default rotation than does a linear update to each Euler angle.

3.4.3 Aligning Edge 3

For the first two image-model feature pairs, solutions exist always despite feature or model measurement error. Errors are "absorbed" by underconstrained degrees of freedom. When the third pair is considered, it is possible for these errors to preclude a perfect solution. Therefore rather than directly employing potentially unstable analytical techniques, which are known to exist for pose computation from 3 features, we use a more robust approach. We use the coarsely sampled intervals of solutions from the first two image-model feature pair constraints as domains for new loci that measure the parallelism of model edge 3 to plane 3.

For each sample, we compute the absolute inner product of edge 3's direction vector with plane 3's normal. These new loci are also well-behaved over their domains. We locate all their local minima and associate model orientations with each one. In practice, these are very few in number. Including locus endpoints, they rarely exceed 10.

To choose among the candidate model orientations, we translate the model to place at least one endpoint of each of its three selected edges in its respective plane. If this translation is possible, we measure the residual distance of the other endpoint from the plane. The average of residuals for a model orientation and translation is taken as an uncertainty measure for that candidate pose. If the translation is impossible for all candidate orientations, then the third image-model feature pair is flagged as incorrect.

Once again, if there are no more image-model feature pairs, then we must choose a target pose on the basis of covering constraints (mentioned below), deviation from prediction, and uncertainty of pose. If there are more image-model feature pairs, then they are incorporated into the pose computation and uncertainty measurement as follows.

For each of the samples used in the previous case, we compute the *average* of the absolute inner products of all the model edges and their respective planes' normals. Thus the local minima for candidate model orientation selection depend on all image-model feature pairs. For translation computation, we place model edges 1 and 2 in their respective planes and compute the average over all other model edges' endpoints of the translation required, along the intersection line of planes 1 and 2, to place the endpoint in its edge's plane of constraint. Thus all image-model feature pairs contribute to the model translation as well. This average of the residual translations serves as the pose's uncertainty measure.

3.4.4 Summary of Perspective Alignment Stage

Resulting pose solutions may be unique, multiple (finitely many), or infinitely many (but maximally constrained), depending on the number and type of constraints provided by the features given. In any case, the Perspective Alignment method finds all solutions, and temporal continuity can be used, along with uncertainty measures, to resolve multiple solutions for tracking purposes. To achieve alignment of each image-model feature pair, we first move the model to the predicted pose if prediction is being used. We then adjust model pose by translations parallel or perpendicular to planes of constraint and unit quaternion rotations about axes of minimal rotation.

Our implementation of this back-projection method is both fast and robust. Real-time results are presented in the next section, and a stability analysis is presented in [Verghese 93]. For brevity, we did not discuss the *covering* constraint, which verifies that every point of the image segment back-projects to a point of the paired model edge. Barring odd collinearities resulting from accidental viewpoints, this constraint is useful for flagging incorrect image-model feature pairings in the early alignment steps.

3.5 Projection for Calibration Stage

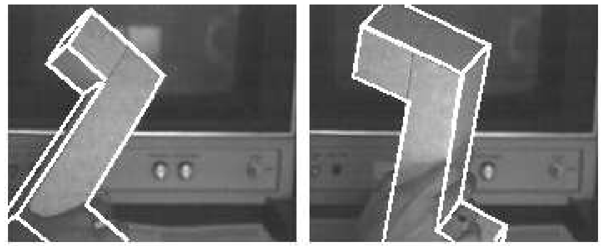
This stage is used only for initial calibration of the feature tracker and for recalibration when features disappear or appear as a result of occlusion or disocclusion. It is not part of the pipeline. Because the feature tracker is a local procedure, there are a number of physical situations that can produce globally inconsistent data. For example, when an object segment first becomes visible because of disocclusion, the feature tracker may incorrectly pair the previously visible model edge with this new object segment. Also, when an approximate pose solution is used, an image segment may be incorrectly paired with a model segment which projects coincidentally.

Local feature tracking must be verified globally with the model. The Perspective Alignment stage flags and ignores any image-model feature pair in the sequence that defies alignment. Since the sequence is ordered by persistence and reliability, the current pair, rather than previous ones, is skipped. Alignment continues with the next pair. If a pair was flagged, the model is projected to recalibrate the feature tracker.

Thus far, this is the only means we have implemented for correcting faulty pairs. Previous tracking methods cannot identify incorrect pairings on an individual basis. Hence there is no known procedure for using this valuable information. Our current prediction process does not anticipate the appearance or disappearance of features. Such preventative methods are under consideration, as are corrective methods for recovering lost pairings without re-recognition.

4. Real Time Results

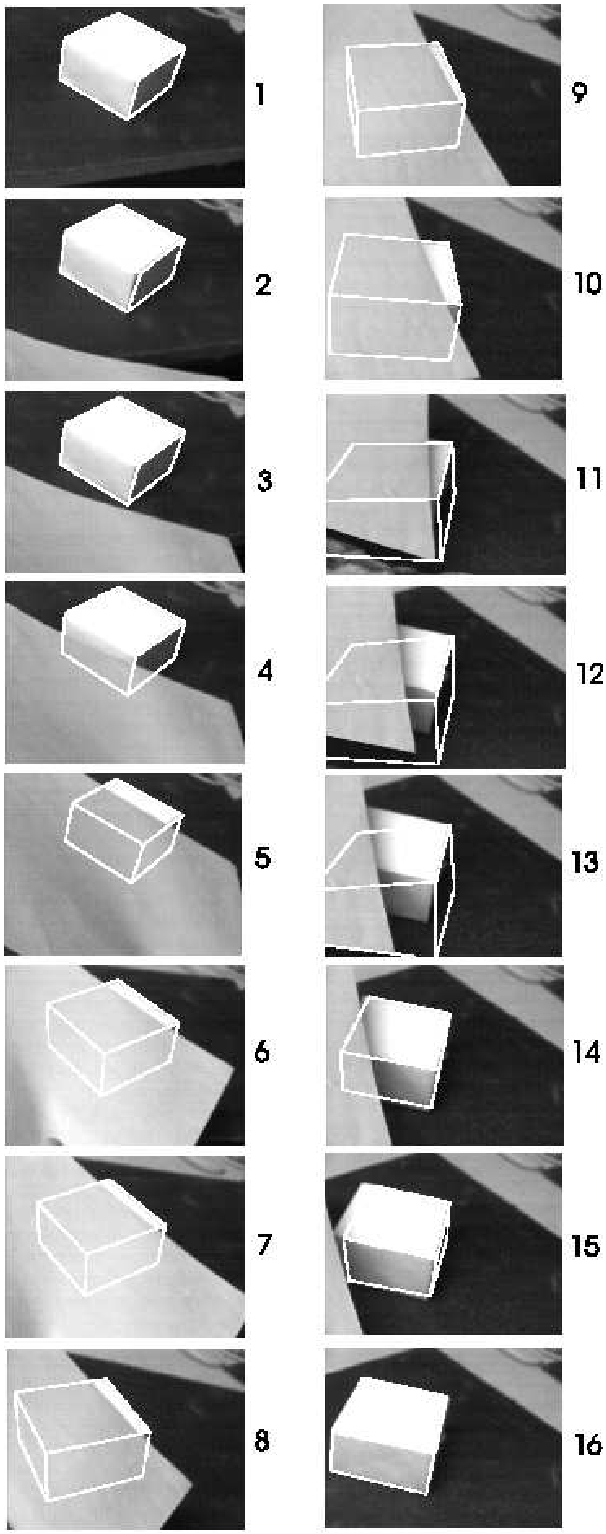
The following frames show a Blocks World Z shaped object being tracked by our tracker. They are snapshots from a real-time motion-tracking sequence. Projected model segments are overlaid with original images to demonstrate model pose maintenance by the algorithm. The object was tracked continuously through several hundred image frames between these two snapshots. (Image sampling requirements are given at the end of this section.) The sequence included several changes in aspect accompanied by appearance and disappearance of object edges. As we discussed, we detect these events in a validation stage, and if there are enough other visible edges, as there were in this example, tracking can continue through aspect changes.



We also performed experiments involving significant occlusion. In several cases, the algorithm recovered completely from near-total occlusion caused by a person's hand holding the object and moving the object randomly. The model remained aligned with the object if its image moved perpendicularly to some visible edge, but since the current version of the algorithm uses the covering constraint only for validation, alignment was lost after extensive object motions by which visible image edges had small perpendicular displacements.

We also positioned an opaque sheet to occlude all but one object edge and then rotated the object. The tracker maintained the visible edge and used it to constrain model pose to help future edge recovery. Recall that model pose cannot be fully determined when only one or two edges are visible. Perspective Alignment's ability to maximally constrain object pose in this case allowed reasonable estimates to be made despite the lack of a global solution.

Results of the latter experiment are shown in the image sequence below. The object is stationary in frames 1 through 5 while the opaque sheet is made to occlude all but one edge. Other tracking methods resort to prediction in this case. Therefore they would not move the model at all and would lose track of the object in frame 6. In frames 6 to 12, the object is rotated through an angle of about 90° with only the single edge visible. In frames 13 to 16 we see the system using newly detected edges to obtain a global solution.



The pipeline stages operate at individual frequencies which may vary over time. As we mentioned, the edge detector always operates at 30Hz on 512×480 8-bit images.

The current bottleneck is the feature tracker. It has an average update rate of about 22Hz for the object shown in this experiment. Its time complexity varies linearly with the number of image segments to be tracked. However, each cycle can detect perpendicular image segment displacement of up to 6 pixels in either direction. Hence an image edge moving up to 132 pixels/s in the image can be tracked. There is a tradeoff between the spatial extent of perpendicular edge strength peak detection and the update rate of the feature tracker. The tradeoff is difficult to analyze since it involves shared image buffer memory contention. Empirical tests have shown that the 12-pixel extent is optimal for the current hardware configuration.

The validation and ordering stage need not operate at a higher frequency than the back-projection stage. Though the validation and ordering stage has the highest computational complexity, *O(nlogn)*, it has the highest maximum frequency since n, the number of image-model feature pairs, is always small. In practice it is synchronized with the feature tracker.

The Perspective Alignment back-projection stage has an update rate of about 10Hz on one of the SGI Power Series processors. Its time complexity is linear in the number of image-model feature pairs used. This compares favourably with locally linear iterative methods, which are $O(n^3)$ [Lowe 91]. However, Perspective Alignment has a high computational constant. Without prediction or filtering, the routine is more than 1000 lines of C code, compared to under 100 lines for plain Newton-Raphson iteration.

5. Conclusions

This paper motivates and demonstrates the use of the Perspective Alignment algorithm for real-time, 3D model-based object tracking. The algorithm was designed primarily to address a shortcoming of all other methods: They do not provide partially complete solutions in underconstrained situations. Partial solutions are essential in order to avoid the computational complexity of re-recognition when attempting to track real-world objects.

New algorithms for sub-problems of the tracking problem were also presented. They include new feature tracking, validation, and calibration processes specifically designed to interact with the Perspective Alignment algorithm.

The method was implemented and demonstrated on a Datacube MV20 image processor connected to a SUN SPARC II workstation and a Silicon Graphics Power Series 4D/380 multiprocessor. Real-time results were shown, with performance and complexity limits provided. The systems kept track of the object in situations that would cause most other real-time trackers to fail.

When occlusion precludes a full solution, and predictions are incorrect, the method maintains a partial pose solution. Additional work is required to take full advantage of Perspective Alignment's ability to maintain partial pose solutions. When additional features become visible, then their pairings with model features should be recovered. More efficient methods than the one presented here can be developed for correctly pairing image features with model features using constraints provided by the partial pose solutions maintained.

Acknowledgements

We would like to thank Brian Down for invaluable assistance in Datacube programming. We acknowledge the support of the Information Technology Research Centre, one of the Province of Ontario Centres of Excellence, the Institute for Robotics and Intelligent Systems, a Network of Centres of Excellence of the Government of Canada, and the Natural Sciences and Engineering Research Council of Canada.

References

[Asada et al 84] M. Asada, M. Yachida and S. Tsuji, Analysis of three-dimensional motions in blocks world, *Pattern Recognition* **17,** 1984, 57-71.

[Bray 90] A.J. Bray, Tracking objects using image disparities, *Image and Vision Computing* **290**, **1** , 1990, 4-9.

[Daucher et al 93] N. Daucher, M. Dhome, J.T. Lapresté, G. Rives, Modelled object pose estimation and tracking by monocular vision, *Proc. British Machine Vision Conference*, 1993, 269-258.

[Dickmanns 90] E.D. Dickmanns, Visual dynamic scene understanding exploiting high-level spatio-temporal models, *Proc. ICPR*, 1990, 373-378.

[Fischler and Bolles 81] M.A. Fischler and R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Graphics and Image Processing* , **24** ,1981, 381-395.

[Gennery 82] D. Gennery, Tracking known three-dimensional objects, *Proc. National Conf. on Artificial Intelligence*, 1982, 13-17.

[Gennery 86] 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.

[Gennery 92] D. Gennery, Visual tracking of known three-dimensional objects, *Int. J. Computer Vision* **7:3** , 1992, 243-270.

[Kent et al 85] E. Kent, M. Shneier and R. Lumia, Pipe — Pipelined image processing engine, *J. Parallel Distrib. Comput.* **2** , 1985, 50-78.

[Haralick et al 91] R.M. Haralick, C. Lee, K. Ottenberg, and M. Nolle, Analysis and solutions of the three-point perspective pose estimation problem, *CVPR*, 1991, 592-598.

[Huttenlocher and Ullman 87] D.P. Huttenlocher and S. Ullman,Object recognition using alignment, *Proc. ICCV*,1987, 102-111.

[Huttenlocher and Ullman 90] D.P. Huttenlocher and S. Ullman, Recognizing solid objects by alignment with an image, *Int. J. Computer Vision*, **5:2** ,1990, 195-212.

[Lowe 85] D. G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer, Boston, 1985.

[Lowe 90] D. G. Lowe, Integrated treatment of matching and measurement errors for robust model-based motion tracking, *Proc. ICCV*, 1990, 436-440.

[Lowe 91] D. G. Lowe, Fitting parameterized three-dimensional models to images, *PAMI* **15 3**, 1991, 441-451.

[Thompson and Mundy 88] D. Thompson and J. Mundy, Motion-based motion analysis: Motion from motion, in *Robotics research: The Fourth International Symposium*, R. Bolles and B. Roth, eds., MIT Press, Cambridge, Mass., 1988, 299-309.

[Verghese et al 90A] G. Verghese, K.L. Gale, and C.R. Dyer, Real-time, parallel tracking of three-dimensional objects from spatiotemporal sequences, in *Parallel Algorithms for Machine Intelligence and Vision*, V. Kumar, P.S. Gopalakrishnan, and L.N. Kanal, eds., Springer-Verlag, New York, 1990.

[Verghese et al 90B] G. Verghese, K.L. Gale, and C.R. Dyer, Real-time motion tracking of three-dimensional objects, *Proc. Robotics and Automation*, 1990, 1998-2003.

[Verghese 93] G.Verghese, Perspective Alignment back-projection for monocular tracking of solid objects, *Proc. British Machine Vision Conference*, 1993, 217-228.

[Wu et al 89] J. Wu, R. Rink, T. Caelli, and V. Gourishankar, Recovery of the 3D location and motion of a rigid object through camera image (an extended Kalman filter approach), *Int. J. Computer Vision* **3**, 1989, 373-394.