Perspective Alignment Back-Projection for Monocular Tracking of Solid Objects

Gilbert Verghese
Department of Computer Science
University of Toronto
6 King's College Road
Toronto, Ontario, Canada M5S 1A5
email: verghese@vis.toronto.edu

Abstract

A major issue in computer vision is the interpretation of three-dimensional (3D) motion of moving objects from a stream of two-dimensional (2D) images. In this paper we consider the problem in which the 3D motion of an object corresponding to a known 3D model is to be tracked using only the displacements of 2D features in the stream of images. There are three main components in any solution to this problem: 2D feature computations, 2D to 3D back-projection, and filtering for noise removal and prediction. In the past five years, most of the research in this area has dealt with filtering and prediction, and some has dealt with 2D feature computations. In this paper, we identify sources of tracking error in the *back-projection* methods and present a new approach which improves tracking confidence. This new back-projection approach, called *Perspective Alignment*, has been implemented, tested, and compared to the conventional locally-linear approximation method. The new algorithm and the results are briefly discussed. Perspective Alignment may also have implications for recognition systems which employ back-projection.

1 Deficiencies in Current Motion-from-Structure Algorithms

A major problem in vision is the computation of a real world object's motion from a stream of images. It is often assumed that fixed structural properties are used to identify an object, and changing state properties are used to determine its motion. The problem of maintaining an accurate representation of a known object's changing state using features found in a stream of images is called *motion-from-structure* [Ullman 79]. In his famous thesis, Ullman noted that the theory of *structure from motion* fails to explain many psychological apparent motion phenomena, and that the motion-from-structure process must be included in any complete theory of motion understanding. Any system for monocular tracking of known 3D objects must solve the motion-from-structure problem.

As the name suggests, motion-from-structure assumes that object shape is known and that relative object motion in space is to be computed. Features denote localizable spatial patterns in 2D images whose frame-to-frame image displacements are assumed computable. Objects denote rigid bodies with six degrees

BMVC 1993 doi:10.5244/C.7.22

of freedom. There is a complex non-linear relationship between projected image feature displacements and 3D object degrees of freedom (motion parameters).

At any instant, the features in an image are formed by the *perspective projection* of real-world object features. However, there is a loss of depth information in the perspective projection process. Therefore the reverse process, back-projection [Lowe 85], is difficult to perform. Nevertheless, under certain conditions, the spatial configuration of a set of image features can fully determine the pose of a model of the object. This is the key to motion-from-structure.

Several motion-from-structure methods use some form of 3D pose hypothesis or voting algorithm whereby best-fitting solutions are chosen (e.g. Normalized Fourier Descriptor Method of [Wallace and Mitchell 80], Clustering Method of [Thompson and Mundy 88], etc.). These methods currently have prohibitive hardware requirements. The image comparisons and Hough transforms used for match measurement are computationally expensive, and hardware requirements depend directly on object complexity. These methods generally have false negatives: The search granularity can cause solutions to be overlooked, as can the subspace projections used in Hough transforms. Results of coarse matching, when used, can be incorrect and therefore a source of false positives. The main problem with all these hit-or-miss algorithms is that they fail to define any measure of model "closeness" to the imaged poses that could serve to direct the search toward the solution.

Other methods use all-or-none back-projection processes based on locally linear systems (e.g. Dynamic Extended Kalman Filter Method of [Wu et al 89], Prediction Error Feedback of [Dickmanns 90], etc.). There must be enough feature information to compute all six degrees of object motion; otherwise the systems soon lose track. They ultimately depend on the success of the Jacobian matrix solution, which is underdetermined when there are not enough features. These methods generally have false positives since locally linear approximations are incorrect when rotations are large. See the following section for results of a stability analysis of the standard iterative back-projection operation.

2 The Perspective Alignment Approach

The existing 3D visual tracking methods lose track of objects unnecessarily. They lose track if some global match method (e.g., image correlation, clustering, matrix solution) fails. However, it may be possible to keep the model's correspondence with one or more local features despite the lack of a global match. The method we propose can drastically improve model-based tracking ability.

We have developed an iterative *perspective alignment* algorithm which determines as many degrees of freedom of object pose as possible using whatever image-model point and line correspondences are available. Point and line correspondences are considered in sequence (e.g., in order of reliability and saliency). Analytic relationships between real-world features and their perspective projections are used to allow each point or line 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. We know of no other tracking system that employs this strategy.

Orthographic alignment of three points [Huttenlocher and Ullman 87] has been used for pruning tree-search in recognition. Orthographic alignment is less com-

plex than perspective alignment: [Huttenlocher and Ullman 90] show that the *orthographic* back-projection problem can be posed as the solution of a *quadratic* polynomial. However, [Fischler and Bolles 81] showed that the solution of a *quartic* is required for *perspective* 3-point back-projection. [Haralick et al 91] compare the various known methods for perspective 3-point back-projection. Their stability analysis turns up singularities and instabilities attributable to the solution of high degree polynomials.

Our perspective alignment solution method does not use a general quartic solver. It directly solves the perspective back-projection problem by a series of restrictions of object pose to a sequence of feature alignment constraints. The perspective alignment algorithms for the special case of edge features are described in this paper. Following that description is a discussion of the results of an implementation of perspective alignment for edge features and a comparison with the results of previous back-projection methods. The local perspective constraints used to develop the alignment algorithms are described first below.

2.1 Local Perspective Constraints from Image Points and Line-segments

Let the camera coordinate frame be (X, Y, Z), with image plane $Z = 1$. A given point feature in the image plane and the camera origin define a 3D ray from the origin in the direction of the image point. In order for the associated model point to align with the image point, the model point must lie along this ray.

See Figure 1a. In order for model point v_1 to align with image point i_1, v_1 must lie on line l_1 formed by the camera origin and i_1. This is the perspective alignment *point* constraint.

See Figure 1b. A given image line-segment feature, if correctly paired with a model edge, arises from the projection of a sub-segment of an object edge. We should align the model in such a way that the model edge's projection covers this image segment. This simple observation gives rise to three useful constraints.

The line-segment in the image plane and the camera origin define a plane in space. The model edge should lie in this plane. If the model edge is parallel to this plane, then the model edge satisfies the *parallelism* constraint. If the model edge intersects this plane, then the model satisfies the *proximity* constraint. The model should also lie in the $Z > 0$ half-space since the object is in front of the camera. Furthermore, if the origin is a general viewpoint, then any ray from the origin through a point of the image line-segment should intersect the model edge. The set of all such rays forms a planar sector which must be *crossed* by the model edge. By this we mean that every point of the image line-segment should back-project to a point of the model edge. This *covering* constraint may be violated by accidental viewpoints and must therefore be used with care. In practice it is usually subsumed by constraints from other features.

Above are the basic geometric constraints used in the perspective alignment algorithm. The algorithm is a decision tree with no backtracking. As we step through the decision tree, we add constraints which further instantiate the model's degrees of freedom. If all the model's remaining 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

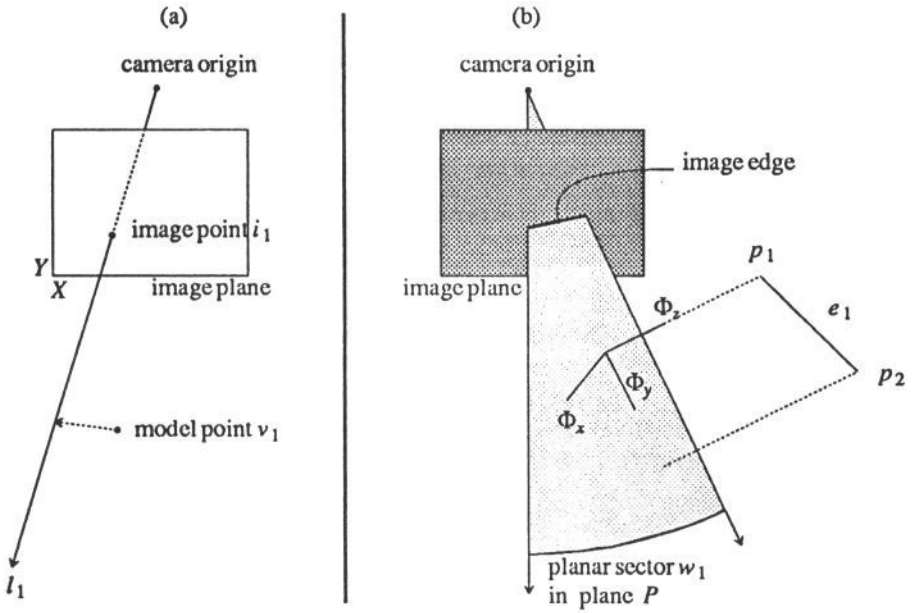

Figure 1: (a) Point constraint. (b) Line constraint.

failure. Previous back-projection methods are incapable of producing such partial solutions.

We will not exhaustively explore the perspective alignment decision tree in this paper. For purposes of comparison to other algorithms it is sufficient to explore the path associated with edges only.

2.2 Perspective Alignment Algorithms

In this section we give perspective alignment algorithms for edge features. There are four cases along the edge-only path of the decision tree. In each case, a new edge is considered, and its constraints are added to the previously satisfied constraints on object pose. Pose is overconstrained in the final case; hence its purpose is to incorporate all additional edges.

One Edge

See Figure 1b. The plane P is defined by an image edge (i.e. line-segment) and the camera origin. In order for this image edge to align with the model edge e_1, e_1 must lie in the plane P and should cross the planar sector. That is, every point of the image edge should back-project to a point of the model edge if the camera origin is a general viewpoint. We define the coordinate frame Φ such that the following are true: $\Phi_{x-y} = P$; the perpendicular projection of model edge e_1

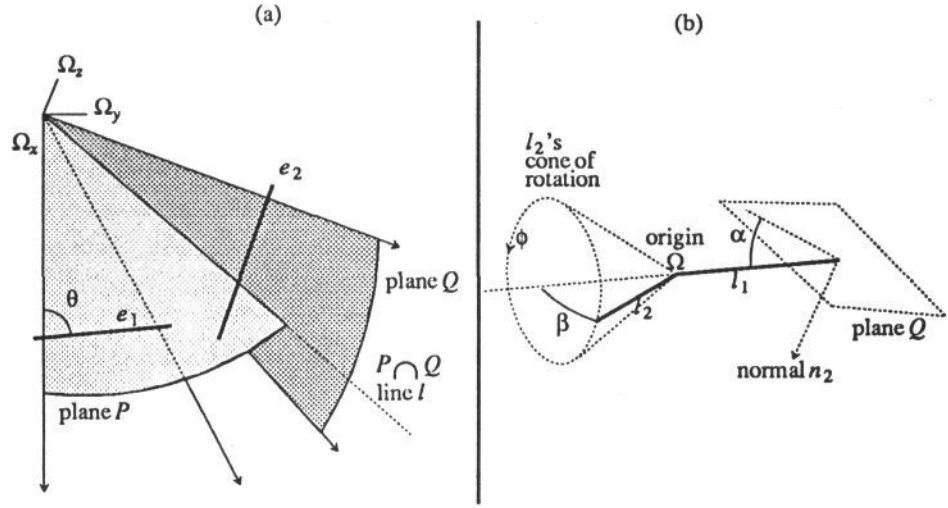

Figure 2: **(a)** Two Edges proximity constraint. **(b)** Two Edges parallelism constraint.

on P lies along the axis Φ_y (if necessary we move the model so that e_1 is not normal to P); and an endpoint p_1 of e_1 lies along Φ_z.

In order to move e_1 into P, we first translate the model by the vector from p_1 to $\Phi(0,\ 0,\ 0)$. We then rotate the model about the axis Φ_x such that p_2 lies on Φ_y. The model can now be rotated and translated in order for e_1 to cross the planar sector. By restricting these model translations to be within P and these rotations to be about axes perpendicular to P, we guarantee that the solution maintains the constraint that e_1 lies in P.

Two Edges

In this case there are two model edges e_1 and e_2, which we must align with their respective image segments. These image segments and the camera origin define planes P and Q. The line l is the intersection of planes P and Q.

We begin by applying the first line-segment's local perspective constraint on the first model edge as described in the single edge case. This places e_1 into P. See Figure 2a. In that case, one rotational and two translational degrees of freedom still exist for this edge, $(\theta,\ \omega_x,\ \omega_y)$. Two constraints will now be used to align e_2 with its image segment: parallelism and proximity.

The parallelism constraint is applied first. For a given value of θ, we determine, as follows, whether it is possible to rotate the model about e_1 to make $e_2 \parallel Q$. Let l_1 and l_2 be the direction vectors of e_1 and e_2, respectively. As shown in Figure 2b, we define α to be the smallest angle formed by l_1 and plane Q (α is the complement of the acute (or right) angle between l_1 and n_2, and though Q passes through the origin, it is shown to the right for clarity). Define β to be the acute

(or right) angle formed by l_1 and l_2. Note that β is fixed by the model's rigidity, while α is determined by θ. It should also be clear by visualizing plane Q at the origin Ω, that $\alpha \leq \beta$ iff the model can be rotated about e_1 to make e_2 parallel to Q. More specifically, if $\alpha < \beta$, then there are two such rotation solutions, ϕ_1 and ϕ_2, and if $\alpha = \beta$, then there is a single such rotation solution, $\phi_1 = \phi_2$. Thus we determine for which values of θ the parallelism constraint is satisfiable.

There are always at least two values of θ for which the parallelism constraint is satisfiable, namely the values for which $\alpha = 0$, i.e., for which e_1 is parallel to Q. The constraint is also satisfiable in two symmetric ranges of values about these two trivial θ solutions. We simply choose the solution which is closest to the model's previous pose.

Now let us consider the proximity constraint for e_2 (Figure 2a). To satisfy this constraint without violating any previous constraints, we must translate e_2 in a direction parallel to plane P. Since the goal is to place e_2 into Q, we choose a direction perpendicular to line l, the intersection of P and Q. If the parallelism constraint for e_2 was satisfiable, then the proximity constraint is also satisfiable.

Thus far, parallelism and proximity constraints were used to align two model edges with their respective image edges. That is, e_1 and e_2 have been placed into planes P and Q. At least two degrees of freedom of model pose remain, leaving us at one of an infinity of satisfactory solutions. For example, translating the model by any vector parallel to line l would not violate any of the constraints used thus far. We could use such a translation to ensure that the model is in front of the camera. We could also use translation to satisfy the covering constraint at this time if we can verify that the origin is at a general viewpoint.

Three Edges

See Figure 3. In this case there are three model edges e_1, e_2, e_3, which must be aligned with 3 image segments that define 3 respective planes with the camera origin, p_1, p_2, p_3. Our alignment algorithm will be: 1) place e_1 into p_1; 2) find model rotations that will make e_2 parallel to p_2 and e_3 parallel to p_3; 3) translate the model so that the edges lie in their respective planes, and measure the goodness of fit; and 4) choose the best fitting solution.

We begin by applying the first line-segment's local perspective constraint on the first model edge as described in the single edge case. This places e_1 into p_1. If we subsequently restrict e_1 to p_1, then the model has two rotational degrees of freedom, θ and ϕ, where θ represents rotation about n_1 (p_1's normal), and ϕ represents rotation about e_1, as shown in Figure 3.

We use these two rotational model degrees of freedom, θ and ϕ, to rotate the model in the second step of our alignment algorithm. For a given value of θ, we can determine as follows whether there are values of ϕ for which $e_2 \parallel p_2$. Let α be the complement of the acute (or right) angle formed by the normals, n_1 and n_2, of the two planes p_1 and p_2. Let β be the acute (or right) angle formed by n_1 and the direction vector of the model edge e_2. If $\alpha < \beta$, then there are two values of ϕ (which coincide if $\alpha = \beta$) for which $e_2 \parallel p_2$. We obtain e_3' and e_3'' by rotating e_3 by θ and then by each value of ϕ. If $e_3' \parallel p_3$ or $e_3'' \parallel p_3$, then a solution for step 2 has been found. However, due to modelling error and noise, there is not likely to be an exact solution. Thus, we search for local minima of the two functions $|e_3' \cdot n_3|$ and $|e_3'' \cdot n_3|$ as functions of θ in the range for which $\alpha \leq \beta$. All these local minima are considered in the next step of the algorithm.

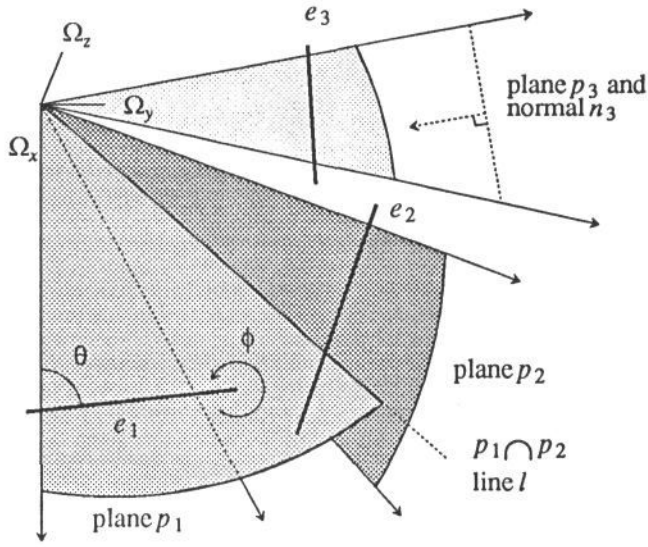

Figure 3: Three Edges

The next step of the algorithm translates the model poses associated with the local minima obtained in the previous step in the two remaining degrees of freedom to attempt to place e_2 in p_2 and e_3 in p_3. These remaining degrees of freedom are translations in the plane p_1. The first translation vector, to place e_2 in p_2, is perpendicular to the line l, while the second translation vector is parallel to the line l. The pose associated with each local minimum from the rotational alignment step is thus translated in two directions to produce a candidate pose solution. These candidate solutions are compared according to the proximity of the projected model endpoints to the image segment endpoints, and finally the best solution is chosen.

More than Three Edges

Three edges are usually sufficient to solve for all six degrees of freedom. However, constraints from additional edges may be "averaged into" the 3-Edge computation as follows. Assume there are N edges, $N > 3$. In the rotational alignment step of the algorithm, rather than using the local minima of the two functions $|e_3' \cdot n_3|$ and $|e_3'' \cdot n_3|$, we use the local minima of the functions $\sum_{i=3}^{N} |e_i' \cdot n_i| \,/\, (N-2)$ and $\sum_{i=3}^{N} |e_i'' \cdot n_i| \,/\, (N-2)$. In the translational alignment step of the algorithm, rather than translating the model parallel to l by the displacement d_3 that places e_3 in p_3, we translate the model parallel to l by the average displacement $\sum_{i=3}^{N} d_i \,/\, (N-2)$.

Use of more than three edges adds slightly to the computation time of the algorithm but adds precision and therefore smoothes the model's tracking motion.

Some special case processing is required when the model edges used are parallel

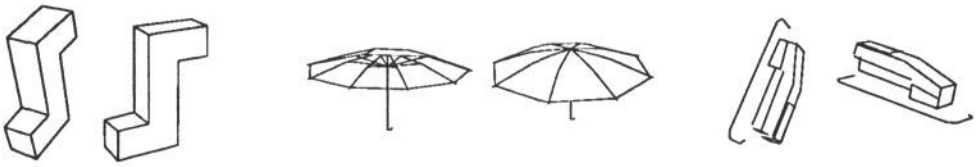

Figure 4: Shown here are two image frames, among thousands used, from the target simulation of each object.

or perpendicular to one another. These anomalies can cause degrees of freedom to appear or collapse and therefore require additional solution processes to those described. A decision sub-tree is used to identify and process all these special cases. They will not be discussed here.

The Perspective Alignment algorithm has proven to be very fast and robust. It is also free of singularities and instabilities. The following section compares it to the conventional iterative back-projection procedure.

2.3 Simulations and Analysis

This section contains results of empirical tests done on the conventional iterative back-projection procedure and on the perspective alignment algorithm. It was assumed that an initial correspondence between the model and the object coordinate systems was known and that any object motion could then occur.

The conventional technique requires computation of the partial derivatives of the 2D line-segment parameters with respect to all viewpoint parameters [Lowe 85]. The technique uses Newton's method to decide how to update model parameters. First the partial derivatives with respect to each of the six model parameters are evaluated at a given projected model endpoint. The sum of each partial derivative times the respective unknown model parameter error correction value is set equal to the point's perpendicular distance to the image line. Each image-model segment pair thus provides two equations (one for each endpoint). This procedure is repeated for three or more image-model segment pairs. Gaussian elimination with row pivoting is used to solve for all six model parameter adjustments. Least squares techniques are used if more than three segments are available. After this system of equations is solved, the model parameters are all updated. This constitutes one iteration of Newton-Raphson convergence.

In order to simulate 2D feature-detected images, wire-frame models were projected and clipped. Typical examples are the two views of a Blocks World S, an umbrella, and a stapler shown in Figure 4.

A random smooth motion sequence was used to simulate the six degree-of-freedom motion of the target object, which was easily trackable by human observers.

A SUN SPARC station 1+ was used to implement and run the simulation and back-projection loops. The loops were synchronized so that one iteration of simulation was followed by one iteration of back-projection for model pose adjustment

by the tracking algorithm. Back-projection was considered unsuccessful only if the iteration diverged. In this case, a counter was incremented to measure tracking errors, the model was immediately reset to correspond with the target, and the simulation continued.

To simulate partial occlusion, we varied the number of features used for back-projection from 7 to 3, and to simulate noise and spatial aliasing, we varied the number of feature coordinate mantissa digits from 5 to 1. The longest projected segments were always selected in order to avoid the instability resulting from short features. The mantissas of the selected image segments' endpoints' coordinates were truncated to the allowed precision, and all other numbers and calculations used double precision. For each of the 25 cases, we recorded the number of tracking errors that occurred over a period of 10 real-time seconds. The results for three objects, the Blocks World S, the umbrella, and the stapler, are shown in the table below:

	Number of Tracking Errors in a 10-second period														
	Blocks World S features					Umbrella features					Stapler features				
dig-	7	6	5	4	3	7	6	5	4	3	7	6	5	4	3
its															
5	0	4	8	49	101	0	0	0	20	23	1	3	21	101	142
4	2	3	18	99	114	0	1	3	28	51	2	11	27	149	182
3	3	5	12	107	131	0	2	4	50	69	7	12	31	157	244
2	3	9	24	100	174	2	2	8	34	72	9	25	46	192	279
1	6	9	23	111	172	2	4	8	52	91	11	27	45	198	298

The variation between objects is due to their varying complexities. Some objects took longer to project, and there were more features for the tracker to choose from. Hence the simulation was slower for these objects, and they had fewer errors overall. The error frequency, i.e., the percentage of back-projection attempts that ended in failure, varied from 0% for the 7-feature-5-digit case to 34% for the 3-feature-1-digit case. The error frequency was largely object independent. Therefore the error count variation across objects is not important.

The variation due to number of features and feature precision is more interesting. For all objects, tracking is reliable only when there are 6 or 7 features and 3 to 5 digits of precision. The most remarkable instability is due to reduction in the number of features. The linear system of equations is underdetermined for any number of features below 3, and although the given numbers of features suffice for pose solution, the results are unreliable for 3, 4 and 5 features, regardless of the features' coordinates' precision. Qualitatively, the model leaps away from the smoothly moving target, more often when there are fewer features, but it follows very smoothly when there are 7 features.

We ran the same simulation for the perspective alignment algorithm. Since the algorithm cannot diverge, we used a more stringent tracking criterion. Failure to align at least 2 features was considered tracking error. Note that alignment of 3 features is often precluded by the simulated spatial aliasing. *The current version*

of the alignment algorithm produces 0 tracking errors in all 75 of the 10-second tracking periods. Qualitatively, the model jitters slightly as it follows the target, more so with less precision, but it never loses track.

The above results clearly indicate that the perspective alignment algorithm is more reliable than the conventional method for tracking 3, 4 or 5 features. The conventional method can produce smoother results when 6 or 7 features are available.

We conducted two additional simulations to compare the algorithms in situations of large inter-frame rotation. Following that, we conducted two more simulations to test the hypothesis that underdetermination may be the main cause of the conventional algorithm's tracking failures.

To compare the algorithms in situations of large inter-frame rotation, we performed essentially the same simulation as above, but each rotation of the target was 30^o. The perspective alignment algorithm had no trouble with large rotations, producing no tracking errors whatsoever. The Newton-Raphson iteration, on the other hand, had an average error frequency of 81% when 3 features were used and 64% otherwise. This might have been expected as a result of the local linear approximation used by the conventional algorithm.

Two more simulations were done to test the hypothesis that underdetermination may be the main cause of the conventional algorithm's failures. Even when 3 or more features are used, it is possible that they do not provide independent constraints on pose. This situation would manifest itself in vanishing pivot elements during the Gaussian elimination. We added code to detect these events and to correlate them with tracking error events. Then we re-ran both the small and large rotation simulations with the conventional algorithm. Underdetermination occurred between 0 and 5 times in each 10-second interval, whereas tracking errors occurred hundreds of times[1]. Therefore most of these errors were not caused by underdetermination. Conventional methods have been combined with filtering to smooth over underdetermination events. However, they occur for extended periods when there is persistent occlusion, or when the image-model feature pairing is incorrect. Filtering is not useful in these situations.

In summary, the conventional Newton method fails for extended periods when few (1-5) features are available or when rotations are large. The perspective alignment method does not fail in these situations. Rather, it produces a maximally-aligned result in case of underdetermination, and it finds a complete pose solution when possible.

3 Conclusions

This paper motivates and introduces the *perspective alignment* algorithm for back-projection. It was designed primarily to address a shortcoming of all other back-projection methods: They do not provide partially complete solutions in under-constrained situations. Partial solutions are essential in order to avoid the computational complexity of re-recognition when attempting to track real-world objects.

[1] Graphical output from this test showed that tracking errors occurred over contiguous intervals of time during the small inter-frame rotation simulation. The conventional method thus has systematic difficulty with certain feature configuration geometries. This phenomenon is a topic of future examination.

Several simulations were run which clearly demonstrate that the perspective alignment algorithm is more robust than the conventional local-linear approximation method for back-projection. In summary, the conventional method systematically fails for certain ranges of orientations when there are few features available, while the perspective alignment method succeeds for all orientations and numbers of features.

The perspective alignment method maintains model correspondence with as few as 1 or 2 image features. The inability to recover lost features is a shortcoming of the current algorithm. When additional features become visible, then their pairings with the correct corresponding model features should be recovered. This ability is expected to raise confidence in perspective alignment tracking systems to a level which was unattainable with previous (all-or-none) back-projection methods.

The implications of perspective alignment for recognition are also under investigation. The usefulness of orthographic alignment for interpretation tree search in recognition has been proven by [Huttenlocher and Ullman 87, 90]. Similar arguments apply to the perspective alignment algorithm. Therefore our work may be viewed as the perspective counterpart to Huttenlocher and Ullman's.

The raison d'être for tracking is to maintain a representation of objects in the world while avoiding the computational complexity of re-recognition. Image-model feature pairing is a combinatorial problem, and each correct image-model feature pair greatly reduces the complexity of the pairing problem. Therefore a tracking system ought to maintain as many pairings as possible if it cannot solve for model pose. Perspective alignment is the first method to achieve this.

Acknowledgements

I would like to thank Veronika Doma for her support and invaluable assistance in typesetting this document, Trinity College Toronto for financial and logistical assistance during the term of my Donship there, and John K. Tsotsos for feedback on previous versions of this paper.

References

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

[Burt et al 89] P. Burt, J. Bergen, R. Hingorani, R. Kolczynski, W.A. Lee, A. Leung, J. Lubin, and H. Shvaytser. Object tracking with a moving camera. In *Proc. Workshop on Visual Motion*, pages 2–12, 1989.

[Crowley et al 88] J. Crowley, P. Stelmaszyk, and C. Discours. Measuring image flow by tracking edge-lines. In *Proc. 2nd Int. Conf. Computer Vision*, pages 658–664, 1988.

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

[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*, 24:381–395, 1981.

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

[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. In *Proc. CVPR*, pages 592–598, 1991.

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

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

[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. In *Proc. ICCV*, pages 436–440, 1990.

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

[Ullman 79] S. Ullman. *The Interpretation of Visual Motion.* MIT Press, Cambridge, Mass., 1979.

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

[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 V. Kumar, P.S. Gopalakrishnan, and L.N. Kanal, editors, *Parallel Algorithms for Machine Intelligence and Vision.* Springer-Verlag, New York, 1990.

[Wallace and Mitchell 80] T.P. Wallace and O.R. Mitchell. Analysis of three-dimensional movement using fourier descriptors. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI*, 2(4):583–588, 1980.

[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). In *Int. J. Computer Vision*, volume 3, pages 373–394, 1989.