Can Too Much Perspective Spoil the View? A Case Study in 2D Affine Versus 3D Perspective Model Matching

J.R. Beveridge and E. Riseman

COINS TR91-86

November 1991

Can Too Much Perspective Spoil the View?
A Case Study in 2D Affine Versus 3D Perspective Model Matching

J. Ross Beveridge E. M. Riseman

Computer and Information Science Department
University of Massachusetts at Amherst
Amherst, MA. 01003 *

November 6, 1991

Abstract

At issue is the importance of fully accounting for 3D perspective in 3D model-based robot navigation. A probabilistic combinatorial optimization algorithm searches for an optimal match between landmark and image features by initiating an iterative generate-and-test procedure from a randomly selected set of correspondence mappings. The 2D-to-2D version of this algorithm approximates full 3D perspective with a 2D affine transform - rotation, translation and scale - applied to a 2D projection of the 3D landmark model. A 3D-to-2D version recomputes the robot's 3D pose relative to the model and reprojects the model during matching. In tests, the 3D-to-2D version reliably recovers the robot's true position. The 2D-to-2D version does equally well when initial errors do not introduce perspective distortion, and does so in roughly one fifth the time. However, it fails on some cases where perspective effects are pronounced.

*This work was supported by the Defense Advanced Research Projects Agency (via TACOM) under contract DAAE07-91-C-R035 and by the National Science Foundation under grant CDA-8922572.

1 Introduction

The problem of matching 3D models to 2D image features arises in many domains. Here we consider problems associated with robot navigation. A robot moving through known surroundings tracks its progress using vision, and moving down a hallway it acquires images such as those shown in Figures 1 and 2. It must test and update its position estimate based upon the appearance of known landmarks. To illustrate, Figures 3A and B show a robot in two possible positions. If the robot believes it is at position A, when it is really at position B, then it should correct this error through landmark recognition.

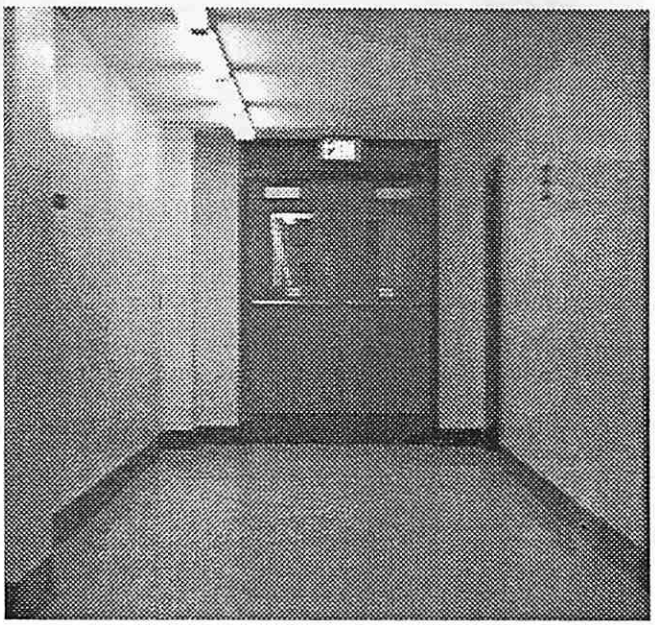

Figure 1: Image 1 taken at Position A

Recognition involves identifying prominent 3D features in an image. In this case, the landmark is a simple, partial, 3D wire frame model of some distinctive features in a hallway. The model can be seen in Figure 3. The image features are 2D line segments extracted from the images using the Burns algorithm [Bur86]. Perspective effects are pronounced in this domain, as Figure 4 illustrates. Figure 4 shows the landmark as it appears to the robot from each of the nine poses shown in Figure 3C. Pose refers to the robot's 3D position and

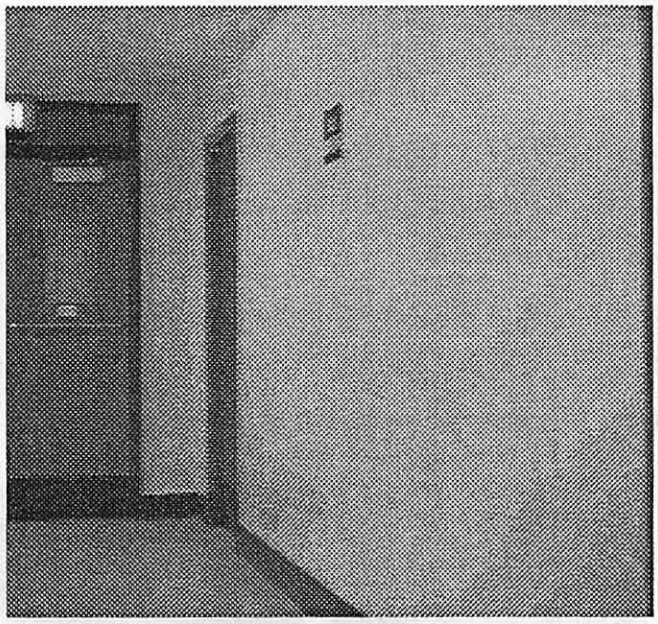

Figure 2: Image 2 taken at Position B

orientation relative to the landmark.

At issue is the importance of accounting for 3D perspective while matching landmark features to image features. In side-by-side tests on a set of example problems, the performance of two versions of an otherwise identical matching algorithm will be compared. The resultant match found by each system is used to estimate the robot's true pose using an algorithm developed by R. Kumar [Kum89; Kum90].

The two versions of the matching algorithm are called the 2D-to-2D system and the 3D-to-2D system. In the 2D-to-2D system, a restricted 2D affine transformation - rotation, translation, and scale in the image plane - is used to account for differences between the landmark's estimated and true appearance. This system matches 2D data to a 2D projection of the model generated from the initial pose estimate. We've described this basic approach previously in [Bev90; Fen90].

In the 3D-to-2D system, the landmark model is repeatedly reprojected into the 2D image

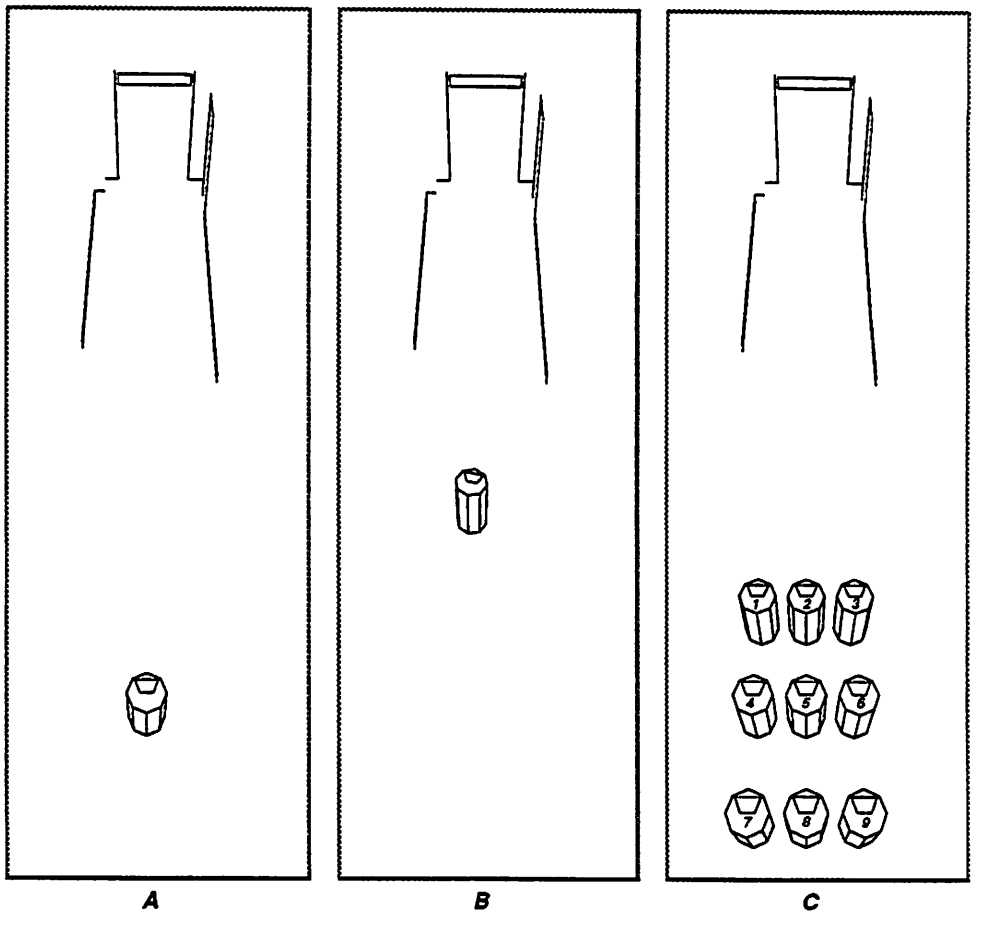

Figure 3: A) Robot in position for Image 1, B) Robot in position for Image 2, C) Nine positions from which to solve for the true position. The 10 3D line segments that make up the landmark are visible in all three examples.

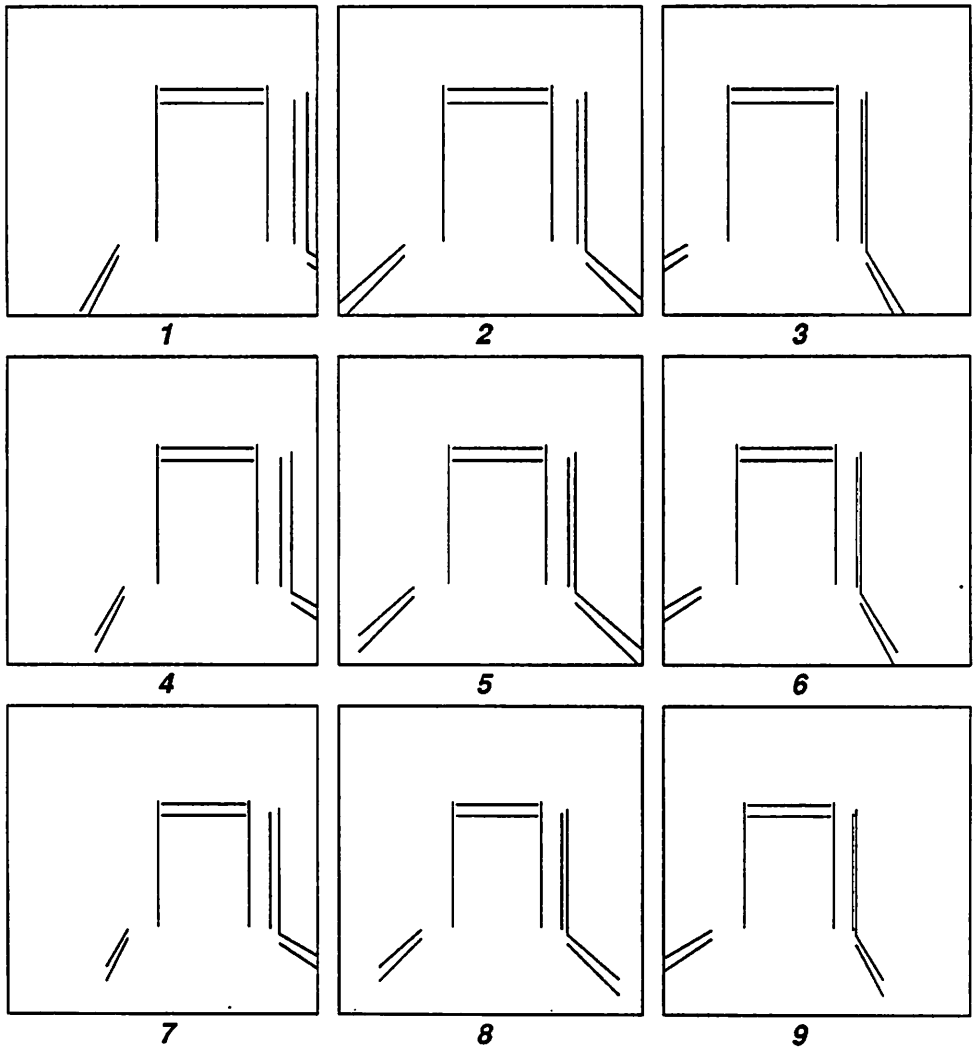

Figure 4: Nine views of landmarks as seen from nine different positions around a nominal location. The relative orientation of the baseboards and the door frame change as the robot's position estimate shifts laterally.

plane based upon incrementally updated 3D pose estimates. This system uses Kumar's iterative pose algorithm [Kum89; Kum90] during matching.

The discrete search space of possible feature mappings is the same for each side-by-side test. The size of these spaces is staggering. To see this, let M be the set of 3D line segments in the landmark model, let D be the set of 2D data line segments in the image, and let S be the set of model-data pairs which are candidate matches.

$$S \subseteq M \times D \tag{1}$$

In landmark recognition, the initial pose estimate usually constrains the possible pairings between model and data segments, and S is considerably smaller than the complete space of pairs, $M \times D$.

It is not prudent to assume that the correct correspondence between model and data features is one-to-one. The process of extracting image features can lead to fragmentation and accretion. We therefore consider many-to-many mappings, and hence the discrete space of possible correspondences C is the powerset of S.

$$C = 2^S \tag{2}$$

Often S contains 50, 100 or more model-data pairs.

The matching algorithm which searches C is a local search algorithm specifically adapted to model matching [Bev89; Bev90]. In general, a local search algorithm moves from an initial solution, via transformations, to one that is locally optimal [Ker72; Lin73; Pap82]. It is certainly not guaranteed to find the globally optimal match. However, if local search is initiated from independently chosen random starting points, the probability of missing the optimum can be made arbitrarily small by sufficiently increasing the number of trials. This random sampling strategy is employed here to find an optimal match.

Optimality is defined in terms of a match error. The optimal match, c^*, minimizes this error function:

$$E_{\text{match}}(c^*) \leq E_{\text{match}}(c) \quad \forall c \in C \tag{3}$$

The match error, E_{match}, is a combination of two terms.

$$E_{\text{match}}(c) = E_{\text{fit}}(c) + E_{\text{om}}(c) \tag{4}$$

The first, E_{fit}, is a residual squared error obtained by first fitting the model to the corresponding data. The second, E_{om}, penalizes matches which omit portions of the model from the match. These two forms of error are discussed below.

The experiments presented here test both the importance and associated cost of accounting for perspective during matching. In experiment 1, both the 2D-to-2D and 3D-to-2D approaches are used to estimate the true position of the robot, position A shown in Figure 3A, from each of the nine initial pose estimates shown in Figure 3C. Experiment 2 tests the ability of each system to recover from a confusion between position A and position B.

2 Landmark Matching and Local Search

In our previous work on landmark-based navigation, [Bev90; Fen90], matching was completely separated from 3D pose recovery [Kum89]. The basic scenario was:

1. Project the 3D landmark model features into the image plane using an initial estimate of the robot's pose. Initial estimates are generated by a navigation/planning module [Fen90].

2. Use a rough estimate of the uncertainty in the initial pose to determine a set of candidate model-data pairs, S defined in equation 1.

3. Use the 2D-to-2D matching algorithm to find an optimal correspondence, c^*, between model and data features [Bev90].

4. Use the corresponding landmark and image features in c^* to recover the 3D pose of the robot relative to the landmark. This is done with the iterative algorithm developed by Kumar [Kum89].

The strength of this approach is that it reduces a 3D-to-2D matching problem to a comparatively simpler 2D-to-2D matching problem. The potential weakness is that rotation, translation and scaling in the plane may be insufficient to recover from an initial error in robot position that produces perspective distortion in the projected 2D model. In hallways, where the side walls come toward the camera, perspective effects associated with lateral position error can be dramatic. Figure 4 illustrates the effect. Note the change in relative angle between the baseboards and the doorway.

For 2D-to-2D matching, a model is fit to corresponding data by solving for the rotation, translation and scale which minimizes the integrated, squared, perpendicular distance between 2D data line segments and infinitely extended 2D model lines. E_{fit} is a normalized function of the residual point-to-line squared error after fitting. The best fit 2D pose has a closed form solution involving a simple quadratic equation [Bev90].

The omission error is a function of the percentage of 2D model line segments not covered by data line segments. Coverage is defined in terms of the perpendicular projection of data segments onto model lines. A point on a model line segment is covered when a point on a data segment projects onto it. A more detailed discussion of the omission error appears in [Bev90].

In extending this approach to 3D-to-2D matching, the 2D-to-2D closed form pose computation is replaced by an iterative 3D-to-2D pose computation developed by R. Kumar [Kum89; Kum90]. This algorithm solves for the 3D position and orientation of the camera relative to a model by minimizing a sum of squared 3D point to plane distances. The points are the endpoints of the 3D model line segments. The planes are defined by the two endpoints of a data line segment and the focal point of the camera. Thus, for the 3D-to-2D system, E_{fit} is a normalized function of the residual point-to-plane squared error after the best fit 3D-to-2D pose has been determined.

The omission error is measured in a manner similar to that used by the 2D-to-2D system. However, it is based upon the projection of the updated model after the best fit 3D-to-2D

pose is generated. The omission is measured relative to the new 2D projections of the 3D model segments.

Two additional points about the implementation of the 3D-to-2D pose algorithm are worth mentioning. First, for a small percentage of cases, the iterative Quasi-Newton method fails to converge to the optimal 3D pose. To overcome this weakness, we use the Levenberg-Marquardt method suggested by David Lowe [Low91]. For a clear summary of this method see [Pre88], pages 542 - 544. Second, to save computation, Kumar's algorithm has been reformulated in terms of state variables associated with each model-data pair, $s \in S$. The sum of these state variables for all s in a particular correspondence c determines the pose. This allows the contribution from a model-data pair s to be added/subtracted from the current sum if s is added/deleted from the current match. This saves considerable amounts of computation during matching, since it removes the the need to loop over the complete set of pairs in c.

3 Inertial-Descent Local Search

The local search matching algorithm employed is a variation of the steepest-descent strategy described in [Bev90]. We call the algorithm *inertial-descent*, and is best explained by example. Figures 5 and 6 illustrate inertial-descent. The matching problem itself is drawn from experiment 2, in which positions A and B are confused (Figure 3). The model and data line segments are shown on the right hand side of Figure 5. The model line segments are shown projected into the image plane as they would appear from position A. The data line segments are derived from the image in Figure 2, acquired at position B . Model line segments are labeled with capital letters, data line segments by number.

Figure 5 also shows a complete trace of two independent runs of the local search algorithm. Each successive row indicates a successively better match. A filled-in square indicates that the associated model-data pair is an element of c. Hence, for example, the first row of

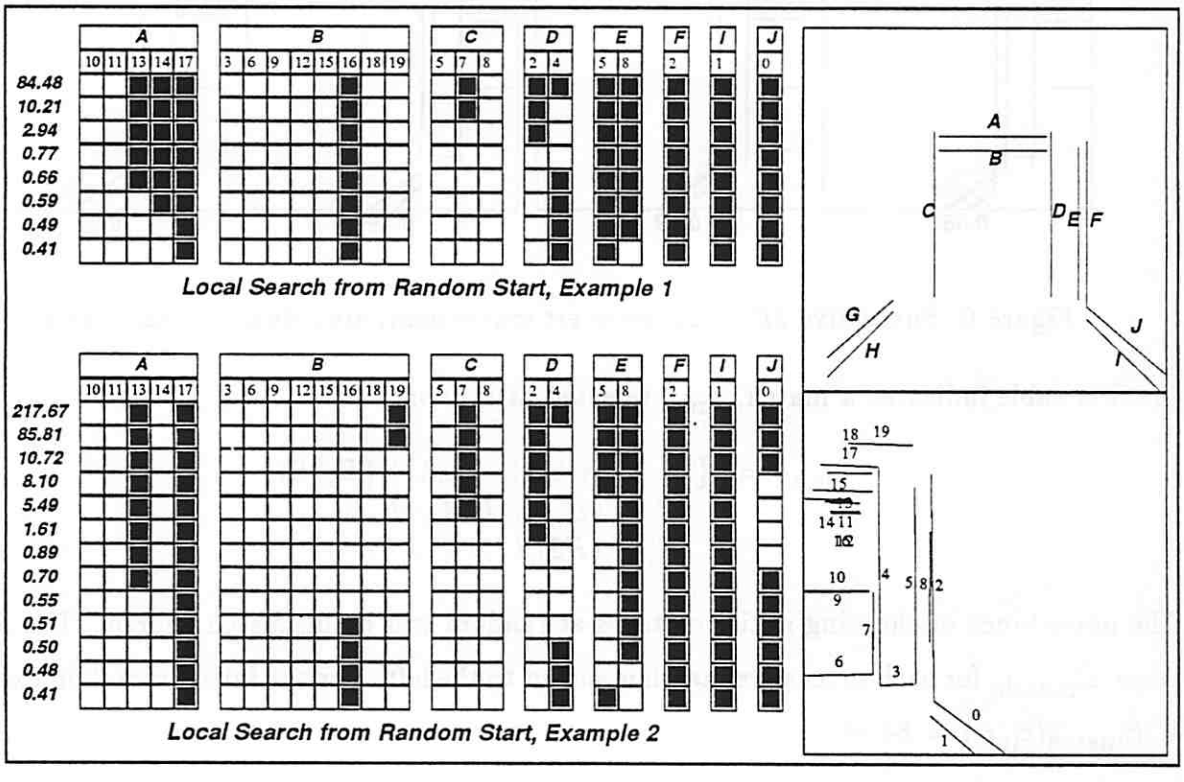

Figure 5: Search space and examples of local search for experiment 2

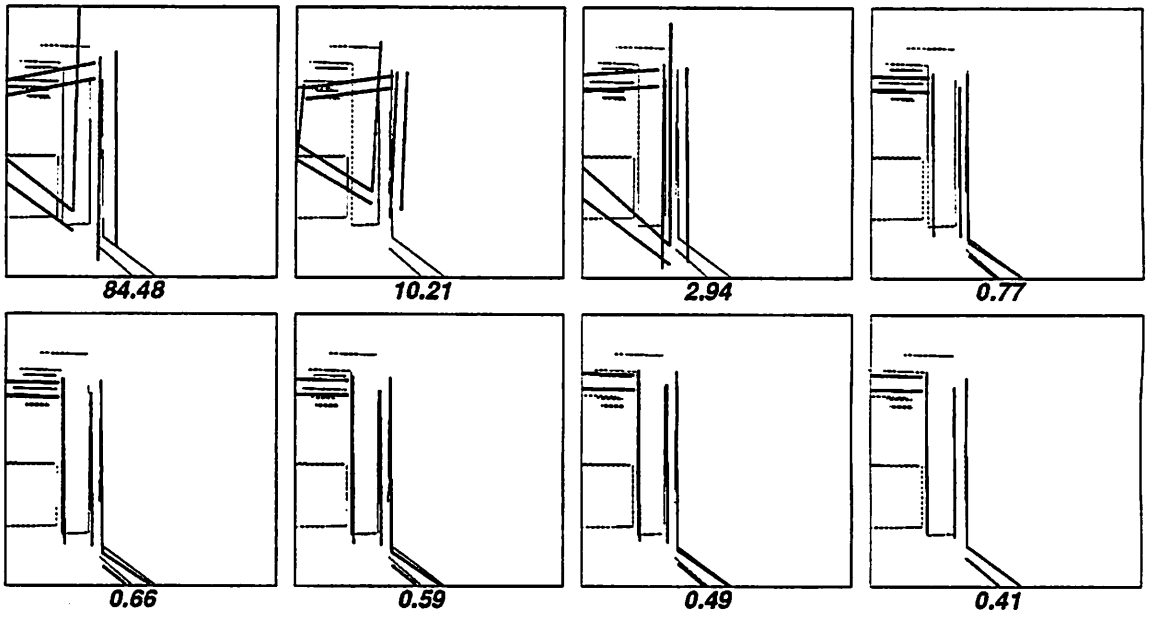

Figure 6: Successive 3D-to-2D pose estimates computed during local search

the first table indicates a match, c_{init}, selected at random.

$$\begin{aligned} c_{\text{init}} \;=\; \{&(A,13),(A,14),(A,17),(B,16),\\ &(C,7),(D,2),(D,4),(E,5),\\ &(E,8),(F2),(I,1),(J,0)\} \end{aligned} \tag{5}$$

The importance of choosing initial matches at random will be discussed shortly. The match error, E_{match}, for each successive match is shown to the left. For the initial match in example 1, $E_{\text{match}}(c_{\text{init}}) = 84.48$.

Each correspondence c may be represented by a bit string. To illustrate, c_{init} from equation 5 may represented as:

$$c_{\text{init}} \;=\; 00111\ 00000100\ 010\ 11\ 11\ 1\ 1\ 1 \tag{6}$$

Spacing is maintained to clarify the relation between the columns in Figure 5 and the bits in equation 6.

Toggling a bit in this string adds or removes a feature pair from the current match. A steepest-descent matching algorithm [Bev90] computes E_{match} for all n correspondences

Match	Improvements in Descending Order
84.48	$((\mathbf{D},4)(D,2)(E,5)(I,1)(J,0)(B,15)(B,12)\ldots$
10.21	$((\mathbf{C},\mathbf{7})(J,0)(I,1)(E,5)(B,12)(B,15)(A,11))$
2.94	$((\mathbf{D},\mathbf{2})(E,5)(E,8)(A,11)(B,12)(B,15))$
0.77	$((\mathbf{D},4)(\mathbf{A},\mathbf{13})(\mathbf{A},\mathbf{14})(\mathbf{E},\mathbf{8})(A,17)(B,15)(B,16))$
0.41	$()$

Table 1: The improvement lists generated by the inertial-descent algorithm each time it tests adding/removing a single pair. The highlighted pairs, applied in sequence, lead to improved matches.

which differ from the current c by one bit, i.e. it tests each single bit toggle. It then applies the best single toggle to the current match yielding a new match c'. After each move, steepest-descent revaluates all n possible toggles. It terminates when no toggle yields improvement. In example 1, (Figure 5), the pair yielding the greatest improvement from the initial match c_{init} is $(D,4)$.

Inertial-descent, like steepest-descent, begins by testing all n single bit toggles. Unlike steepest-descent, rather than just applying the single toggle yielding the greatest improvement, inertial-descent builds a sorted list of all toggles yielding improvement. It then works down this list sequentially toggling successive pairs until either: the list is exhausted, or a toggle is found which no longer produces an improvement.

Table 3 illustrates the improvement lists generated by the inertial-descent algorithm for example 1 (Figure 5). The highlighted pairs at the head of the lists actually lead to improved matches. For the first three matches, inertial-descent didn't save any computation relative to steepest-descent. From the new match obtained by toggling the first pair on the list, the second pair no longer improved the match, and therefore all n toggles were again tested. However, for the fourth match, the list allowed the algorithm to apply four toggles in succession without expending effort testing alternatives. In general, inertial-descent tests far fewer matches than steepest-descent.

As already mentioned, each time the local search algorithm tests a new correspondence c, it must compute $E_{\text{match}}(c)$. To emphasis that new 3D poses are generated for each match tested by the 3D-to-2D system, Figure 6 shows the projection of the landmark from these updated poses for the successively better matches found in example 1, Figure 5.

The local search algorithm just illustrated may not find the globally optimal match c^*. However, by running multiple trials from randomly selected initial matches, and then taking the best match found in the series of trials, a local search procedure with a relatively small probability of finding the globally optimal match on a single trial may be used to find the global optimum with very high probability.

Formally, let P_s be the probability of successfully finding the global optimum on a single trial. The conjunctive probability of failing to find the global optimum in t independent trials is Q_f:

$$Q_f = P_f^t \qquad P_f = 1 - P_s \tag{7}$$

Therefore, the number of trials required to find the global optimum with probability Q_s, using a local search algorithm with probability of success P_s, is given by the following equation.

$$t = \lceil \log_{P_f} Q_f \rceil \qquad Q_f = 1 - Q_s \tag{8}$$

To illustrate, if $P_s = 0.10$ then 29 trials are required to find the optimum with probability $Q_s = 0.95$. If $P_s = 0.5$ then only 5 trials are required to reach the same confidence level.

Empirical trials on the landmark recognition problems presented below are used to estimate the true probability of success, P_s, and hence the number trials, t_s, required to obtain the optimal match with confidence 95% or better. In general, 100 trials of local search are used to generate estimates $\hat{P}_s$ and $\hat{t}_s$.

The estimate $\hat{P}_s$ is subject to one important qualification: it may not be based not upon the true globally optimal match c^*, but instead upon what we'll call the optimal match, the best match found in a series of empirical trials. By visual inspection one can often conclude that the optimal match and the globally optimal match are the same. However, this is not

always the case.

4 Experiment 1

The task is to recover the true position of the robot from each of the nine pose estimates shown in Figure 3C. [1] The true position, position A, is 41.3 feet from the doorway and 4 feet from each of the two side walls. The nine test positions were obtained by translating the true position estimate forward and backward and side-to-side. Estimates $1-3$ are 5.3 feet forward of position A. Estimates $4-6$ are 1.3 feet forward of A, and estimates $7-9$ are 3.7 feet back of A. Estimates 1, 4 and 7 are 2 feet to the left of A. Estimates 3, 6 and 9 are 2 feet to the right of A. The landmark model as it appears from these nine positions has already been shown in Figure 4.

In addition to varying the initial pose estimates, this experiment considers both 'directed' and 'undirected' model segments. A directed segment specifies the sign of the intensity gradient across the edge. By experimenting with search spaces generated both with and without directed segments we show that using directed segments saves computation. However, at least for the 3D-to-2D system, the quality of the final match and the associated updated pose estimate is the same in each case.

The final pose estimates generated by the 3D-to-2D system are within 0.1 feet of the true pose for all nine initial estimates and for both the directed and undirected search spaces. Essentially the same match is found in all cases, which explains the consistently good final pose estimate. The full 3D perspective approach appears to be very robust.

Figure 8 shows the model and data line segments. The model is projected from the true position. The optimal match, c^*, found by the 3D-to-2D system for the directed search space

[1] In this experiment we only consider the position portion of the pose estimate associated with a match.

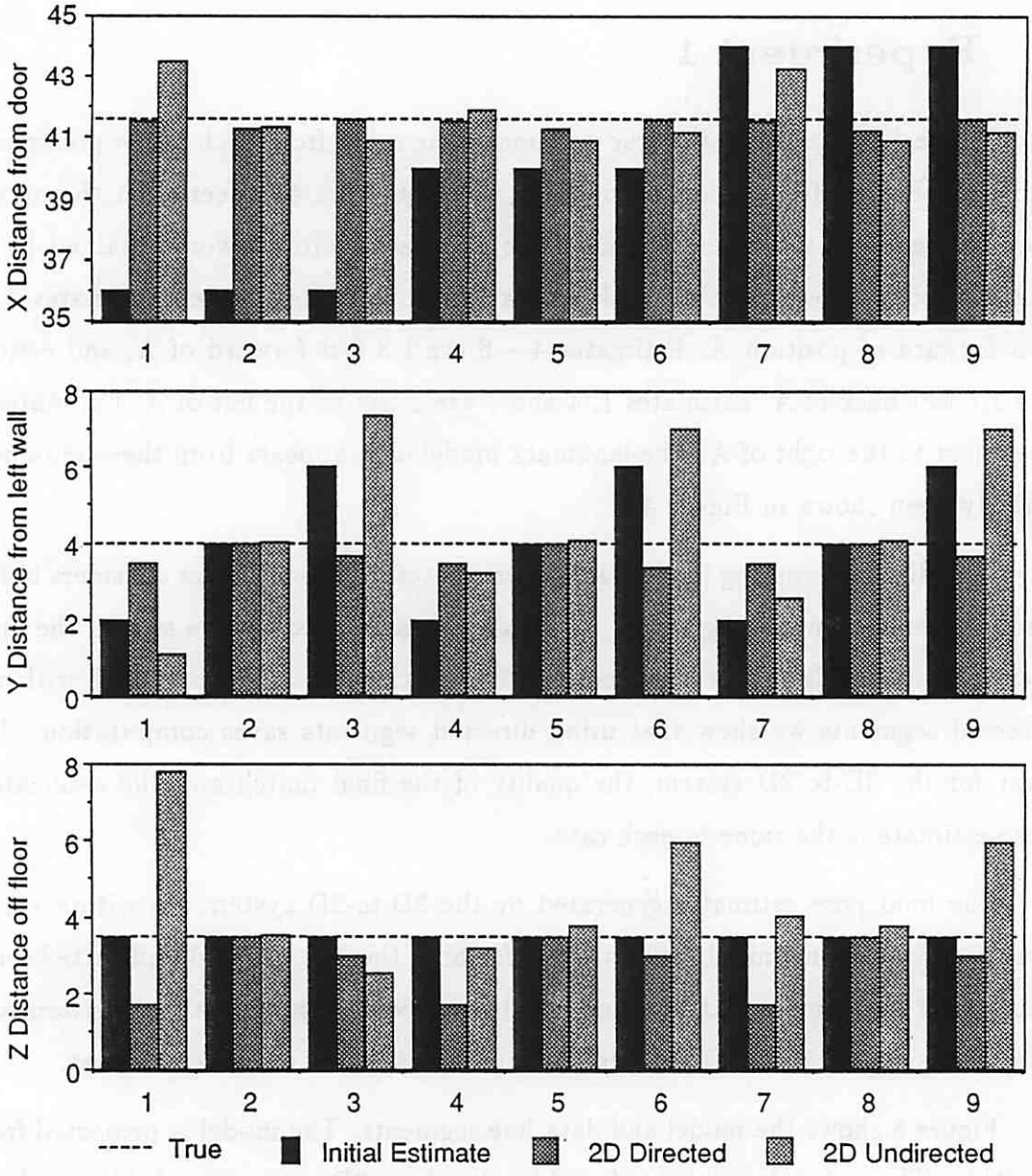

Figure 7: Bar chart of position estimates recovered with 2D-to-2D matching. Position is shown along each of the three dimensions for the best matches. Results for the directed and the undirected search spaces are shown.

is:

$$c^* = \{(A,32),(A,33),(B,30),(C,15),(D,12),(E,14),(F,11),(G,10),(H,9),(I,7),(J,0)\} \quad (9)$$

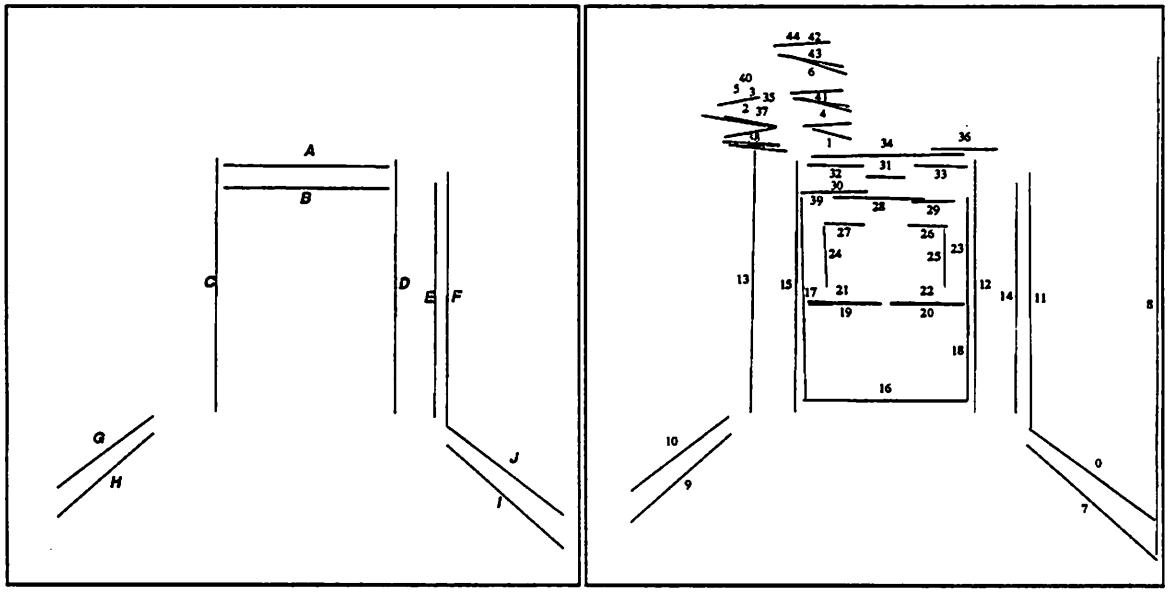

Figure 8: Labeled landmark and data line segments for Image 1

The final pose estimates for the 2D-to-2D system are presented in Figure 7. For cases 2, 5 and 8, for both the directed and undirected search spaces, the 2D-to-2D system recovers the robot's true position essentially as well as the 3D-to-2D system. This is to be expected, since forward and backward error primarily changes the expected scale of the landmark model. However, for the other cases the recovered pose estimates are not as good. For the directed search space the final position is always better than the initial estimate. However, for the undirected search space this is not always the case. In particular, for cases 3, 6 and 9 the recovered Y position is worse than the initial estimate.

The first step in generating these results was to determine the set of candidate pairs S. The selection of candidate pairs depends upon the placement of the 2D projection of a model line m in the image. For the case of directed model segments, a pair $s = (m,d) \in M \times D$

	Initial Pose Estimate								
	1	2	3	4	5	6	7	8	9
$\|S_d\|$	41	36	42	45	37	42	53	43	54
$\|S_u\|$	87	75	89	94	77	94	112	92	112

Table 2: The number of candidate pairs for each of the nine initial pose estimates and using directed, S_d, and undirected, S_u model segments.

is an element of S_d if:

1 d is within 30 degrees of m.

2 d is within 128 pixels of m.

3 d is at least 1/4 the length of m.

4 d and m have the same sign of contrast.

For the undirected case, S_u, the sign of contrast test is omitted. These bounds are picked based on experience with the domain. In particular, 128 pixels is one quarter the distance across the full 512 by 512 image and is adequate to ensure the correct match is contained in the resultant search space. The number of candidate pairs in S_d and S_u for each of the nine pose estimates are summarized in Table 4.

We tested two alternative ways of generating the initial random correspondences used as starting points by the local search algorithm. In one case, an initial correspondence c_{init} was selected uniformly from the complete search space C. In the other, we biased the selection to favor choices of c_{init} with roughly 2 data segments bound to each model segment.

This is done by defining a binding probability, $P_B(s)$, such that a pair $s = (m, d)$ is included in an initial correspondence c_{init} with probability $P_B(s)$. Choosing $P_B = 0.5$ for all pairs yields the uniform sampling mentioned above. Defining $P_B(s)$ as follows biases selection. Let k_m be the number of pairs in S which include model feature m, and then

define

$$P_B(s) \;=\; \max(0.5, 2/k(m)) \tag{10}$$

The maximum of 0.5 is desirable because it randomizes cases where there are only 1, 2 or 3 candidate pairs for a model segment.

Results comparing uniform random versus biased random selection are presented in Figure 9. These results are for the directed search space. Both systems were run 100 times for each of the nine position estimates. Figure 9 shows the number of times the optimal match was found in each case. These outcomes suggest the true probability of success, P_s, is higher for biased random selection, and that biased selection is superior to uniform selection. The biased selection defined by equation 10 is therefore used henceforth.

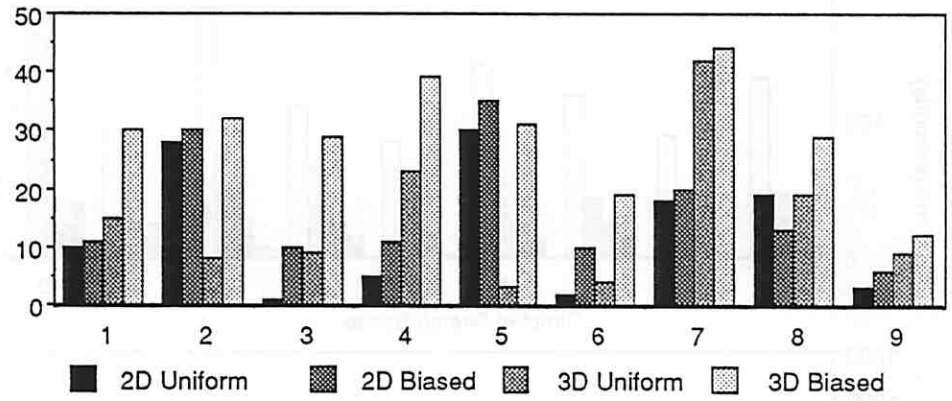

Figure 9: Trials out of 100 yielding the optimal match for uniform verses biased selection of initial feature bindings.

The true probability of success, P_s, is a parameter of a binomial - success/failure - process. The maximum likelihood estimate, $\hat{P}_s$, is just the ratio of the number of times the optimal match is found over the total number of trials. Plugging $\hat{P}_s$ into equation 8 yields the estimated number of trials, $\hat{t_s}$, required to find the optimal match with confidence 95% or better. The graph in Figure 10 compares $\hat{t_s}$ for the 3D-to-2D and 2D-to-2D systems over all nine initial pose estimates. These results are for the directed search space. Observe that $\hat{t_s}$ for the 2D-to-2D and 3D-to-2D systems are more similar for cases 2, 5 and 8. Also note

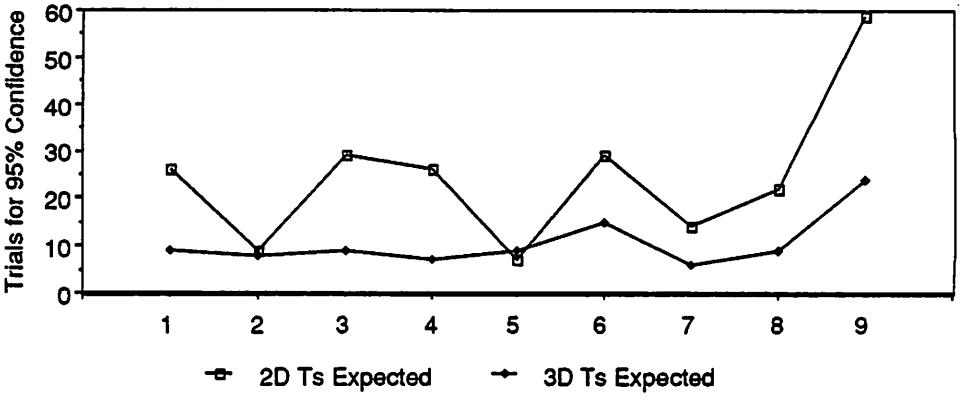

Figure 10: The number of trials required to find the optimal match with 95% confidence

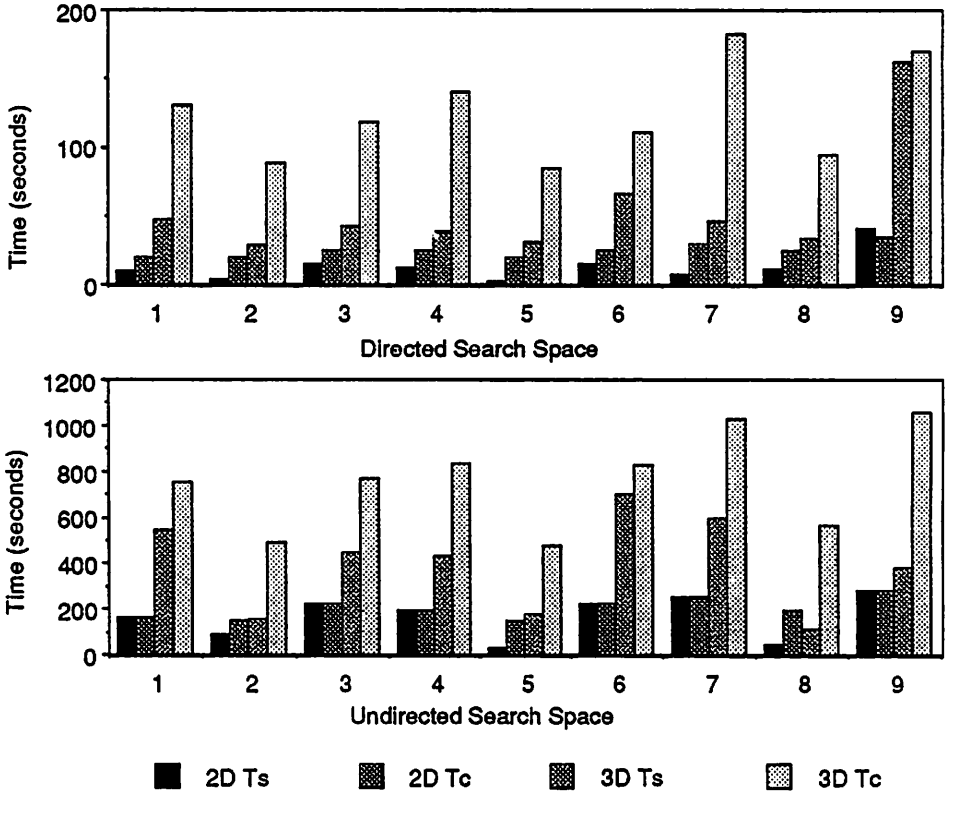

Figure 11: Timing information in seconds for a TI Explorer II Lisp Machine. For both the directed and undirected search spaces the 2D-to-2D system runs in roughly 1/5 the time required by the 3D-to-2D system.

the 3D-to-2D system generally requires fewer trials.

The expected amount of time required to find an optimal match is the expected amount of time to run a single trial times the expected number of required trials. Timing information for the 2D-to-2D and 3D-to-2D systems is presented in Figure 11. All times are reported in seconds and are for a TI Explorer II Lisp Machine. Two numbers are given for each case, the first is the number of seconds required to run $\hat{t}_s$ trials. The second number is a more realistic estimate, based upon the time required to run a conservative number of trials, t_c. A conservative number is selected which essentially guarantees finding the optimal match with confidence 95% across every one of the set of problems.

For the directed search space and the 2D-to-2D system, $t_c = 50$ is chosen. Since $\hat{P}_s$ is consistently higher for the 3D-to-2D system, a value of $t_c = 25$ is chosen for this system. For the undirected space the 3D-to-2D system finds the optimal match less often, and $t_c = 50$ is necessary.

For the undirected space and the 2D-to-2D system the definition of success is extended to include the best three matches found. This is necessary because for all but cases 2, 5 and 8 the probability of finding the single uniquely best match becomes very low. Making this change, and running 300 trials, $\hat{P}_s \geq 0.02$. Under these conditions, $t_c = 150$. To be conservative, the worst pose generated by the top three matches was reported in Figure 7.

5 Experiment 2

Consider confusing position A with position B (Figure 3). For this problem, Table 3 shows the true positions, initial pose estimates, and recovered pose estimates. The 3D-to-2D system recovers the true pose to within 1 foot in both cases. However, in recovering position A from an initial estimate of position B, *Est-B-True-A*, the 3D-to-2D system found the best match only once in 300 trials. The next best match was almost equally good, and was found in five out of 300 trials. Success in this case is redefined as finding one of the two best

matches, $\hat{P}_s = 0.02$. The recovered pose for the two best 3D-to-2D matches is shown for this case. The 2D-to-2D system did less well, improving the initial estimate in each case, but still missing the true position by several feet.

Est-A-True-B				
	X	Y	Z	Δ
True	30.0	4.0	3.5	
Estimate	40.0	4.0	3.5	10.0
2D	27.0	5.1	2.8	3.3
3D	30.3	3.7	3.9	0.6

Est-B-True-A				
	X	Y	Z	Δ
True	41.3	4.0	3.5	
Estimate	30.0	4.0	3.5	11.3
2D	46.2	1.6	5.6	5.5
3D	40.7	3.8	3.8	0.7
$3D^e$	41.1	3.6	3.9	0.6

Table 3: Pose results when positions A and B are confused. The 3D-to-2D system recovers pose well, but in the *Est-B-True-A case* two almost equally good matches are found, noted as $3D$ and $3D^e$. The 2D-to-2D system does less well.

The estimated probability of success, $\hat{P}_s$, and required number of trials, $\hat{t}_s$, was determined for both systems on both problems. Based upon $\hat{t}_s$, a conservative number of trials t_c was selected, and finally the expected time required to run t_c trials determined. These results are presented in Table 5. As before, times were measured on a TI Explorer II Lisp Machine.

	Est-A-True-B		Est-B-True-A	
	2D-to-2D	3D-to-2D	2D-to-2D	3D-to-2D
$\hat{P}_s$	0.11	0.22	0.08	0.02
$\hat{t}_s$	26	13	36	150
t_c	50	150	50	150
seconds	10	585	35	1,710

Table 4: The estimated number of required trials, $\hat{t}_s$ for Experiment 2. Also the conservative number of trials, t_c and the time required to run this many trials.

6 Conclusion

These experiments provide insight into the importance of perspective. We've compared 2D-to-2D matching with 3D-to-2D matching under conditions where the 2D-to-2D approach might be expected to fail. The results suggest that the 2D-to-2D approach is more useful and reliable than one might at first expect. The results also suggest that the additional cost of doing full 3D-to-2D matching is not prohibitive, and a prudent system might choose to always employ 3D-to-2D matching.

Hybrid algorithms, which blend 2D-to-2D and 3D-to-2D matching, present intriguing possibilities for the future. It is worth noting that the 2D-to-2D system usually improved errorful pose estimates. Hybrid algorithms might well recover 3D-to-2D matches as reliably as the full 3D-to-2D system used here, but with run times closer to those shown for the 2D-to-2D system. One promising hybrid might use 2D-to-2D matching initially and switch to 3D-to-2D matching only after a 2D-to-2D optimal match has been found. Another variation might use the comparatively cheaper 2D-to-2D test to determine a next candidate move, double checking the move by computing a new 3D pose and reprojecting the model.

References

[Bur86] J. B. Burns, A. R. Hanson, and E. M. Riseman. Extracting straight lines. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI–8(4):425 – 456, July 1986.

[Bev89] J. Ross Beveridge, Rich Weiss, and Edward M. Riseman. Optimization of 2-dimensional model matching. In *Proceedings: Image Understanding Workshop*, pages 815 – 830, Los Altos, CA, June 1989. DARPA, Morgan Kaufmann Publishers, Inc (Also a Tech. Report).

[Bev90] J. Ross Beveridge, Rich Weiss, and Edward M. Riseman. Combinatorial optimization applied to variable scale 2D model matching. In *Proceedings of the IEEE*

International Conference on Pattern Recognition 1990, Atlantic City, pages 18 – 23. IEEE, June 1990.

[Fen90] Claude Fennema, Allen Hanson, Edward Riseman, J. R. Beveridge, and R. Kumar. Model-directed mobile robot navigation. *IEEE Trans. on Syst., Man, Cybern.*, 20(6):1352 – 1369, November/December 1990.

[Kum89] Rakesh Kumar and Allen Hanson. Robust estimation of camera location and orientation from noisy data having outliers. In *Proc. of IEEE Workshop on Interpretation of 3D Scenes*, pages 52 – 60, Austin, TX, 1989. IEEE.

[Kum90] Rakesh Kumar and Allen Hanson. Analysis of different robust methods for pose refinement. In *Proc. of IEEE Workshop on Robust Methods in Computer Vision*, pages 161 – 182, Seattle, WA, 1990. IEEE.

[Ker72] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Tech. Journal*, 49:291 – 307, 1972.

[Lin73] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498 – 516, 1973.

[Low91] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(5):441 – 450, May 1991.

[Pre88] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1988.

[Pap82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, chapter Local Search, pages 454 – 480. Prentice–Hall, Englewood Cliffs, NJ, 1982.