Wide-Area Egomotion Estimation from Known 3D Structure

Olivier Koch
koch@csail.mit.edu

Seth Teller
teller@csail.mit.edu

MIT Computer Science and Artificial Intelligence Laboratory
Stata Center, 32 Vassar Street, Cambridge MA 02139

Abstract

Robust egomotion recovery for extended camera excursions has long been a challenge for machine vision researchers. Existing algorithms handle spatially limited environments and tend to consume prohibitive computational resources with increasing excursion time and distance.

We describe an egomotion estimation algorithm that takes as input a coarse 3D model of an environment, and an omnidirectional video sequence captured within the environment, and produces as output a reconstruction of the camera's 6-DOF egomotion expressed in the coordinates of the input model. The principal novelty of our method is a robust matching algorithm that associates 2D edges from the video with 3D line segments from the input model.

Our system handles 3-DOF and 6-DOF camera excursions of hundreds of meters within real, cluttered environments. It uses a novel prior visibility analysis to speed initialization and dramatically accelerate image-to-model matching. We demonstrate the method's operation, and qualitatively and quantitatively evaluate its performance, on both synthetic and real image sequences.

1. Introduction

Robust, wide-area egomotion estimation within general environments is one longstanding goal of computer vision researchers. Existing vision-based SLAM (Simultaneous Localization and Mapping) or SFM (Structure From Motion) algorithms expend storage and computational resources that grow super-linearly with the sequence length, and incur growing localization error over time; these methods typically handle only short-duration, short-excursion sequences [9, 23].

This paper describes an alternative approach to vision-based localization which assumes availability of a coarse 3D environmental model before exploration commences, rather than constructing the model on the fly. We show that under these circumstances egomotion estimation can be made sufficiently robust and efficient for real-time use with extended camera excursions through multiple buildings. Quantitatively, our method can recover the 6-DOF rigid body pose of a camera (attached to a user's head, body, or hand-held device) as it is moved within a spatially extended, visually cluttered environment, with an accuracy of a 10-25 cm in translation and about two degrees in orientation. Our algorithm has been tested over dozens of minutes of walking-speed motion within an interconnected collection of buildings with many corridors and hundreds of distinct rooms.

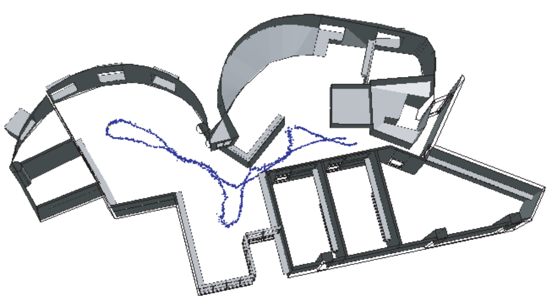

Figure 1. Recovered motion of an omnidirectional camera during a long excursion (1,500 frames) within an extended cluttered environment (450 m^2). Our method establishes the camera's initial location, then tracks the camera as it moves.

Capturing high-fidelity CAD models of existing built environments is itself a difficult problem, and in some respects our solution eases one hard problem (egomotion estimation) by assuming the solution to another (as-built model capture). Yet we believe that this is a useful tradeoff, for two reasons. First, 3D CAD models of existing spaces will become more commonly available as GIS and other mapping efforts extend indoors and users grow to expect the same map coverage and quality indoors that they currently enjoy for outdoor spaces and road networks. Second, our method requires for effective operation only *coarse* model geometry, comprising only major visible building elements, typically walls, floors, ceilings, doors and windows. In many cases such models can be generated through automated "extrusion" of 2D floorplans (as is the case with most 3D model data used in this paper).

We formulate egomotion estimation as an on-line task alternating between two operating phases. The *Initialization* phase determines a valid camera pose estimate when the camera pose is known poorly, either at the start of exploration or after "loss of lock." Once the Initialization phase establishes an accurate camera pose estimate for one or more frames, the *Maintenance* phase updates camera pose over subsequent frames. We show how both phases can be dramatically accelerated through prior *Visibility Analysis* of the environment model.

Our system makes four significant assumptions. First, we assume that a coarse polyhedral model of the environment – which we define as including, at a minimum, walls, floors, ceilings, doors and windows – is supplied as input. Such a model could be provided by the building's architects, or produced independently by a post-construction modeling method. However it is acquired, we extract 3D model "segments" from the boundaries of each polygon in the input model. Second, we assume that the camera is intrinsically calibrated. Third, we assume that camera motion is smooth, *i.e.* that sensor-dependent linear and rotational velocity bounds are not exceeded. Finally, we assume that the camera never moves through any impenetrable, opaque surface, and consequently can never observe the back side of any polygon in the input model.

We emphasize that we do not make a number of other assumptions found in other vision-based localization systems. For example, we do not assume the presence of vertical and horizontal model edges [8], right angles [3], or vanishing points [6]. We do not assume knowledge of surface color or reflectance attributes in the environment, or indeed of any "appearance" information other than knowledge of the geometric model itself. Finally, though we do assume that the portions of the environment represented by the provided model are static, we do not assume a static world. In particular, our method handles time-varying lighting, time-varying clutter (e.g. furniture), and transient image motion (caused e.g. by passers-by).

2. Related Work

The theoretical background of vision-based localization is presented in two seminal books [16, 12] and a more recent survey of multi-view geometry [15].

Three line correspondences are sufficient in theory to recover 6-DOF camera pose [10] though in practice more lines may be required [21, 1]. Point features and RANSAC may be combined [19] to achieve robust real-time localization. In general, standard methods operate by tracking point, edge, or contour features between consecutive frames [18, 11, 14], and minimizing some error function. Other researchers have combined point-based and segment-based tracking methods for increased robustness [22].

Alternatively, the geometry of the environment may be reconstructed explicitly [24], with an optimization based on Plücker coordinates allowing the removal of superfluous degrees of freedom [4, 5]. Our method requires no training phase (as in [20]), no artifical landmarks (as in [17]) and one omnidirectional camera (rather than two as in [7]).

3. Contributions

Our method differs from existing work in four respects. First, it uses omnidirectional images in order to support full view freedom (*e.g.*, close proximity to environment surfaces), and to remove pointing constraints from the camera operator. Second, the method scales to large, real-world environments (see section 5). Third, the method includes an automated initialization capability which runs more efficiently when the user provides a "hint" about the camera's location. Finally, the method is robust to significant clutter, lighting variations, and transient motion.

4. Egomotion Estimation Method

Our method consists of matching detected 2D image edges to known 3D model segments, without performing structure-from-motion. We chose to base our egomotion estimation on line tracking, rather than point tracking, both because this approach seemed relatively unexplored in the vision literature, and because intuitively we expected long model segments to be robustly detectable and precisely localizable even in the presence of severe clutter.

Given a set of correspondences between image edges and model segments, we recover the camera pose by minimizing an error function ξ, defined as the normalized square sum of angular disparities for each correspondence between image edge and (reprojected) model segment (Figure 2):

$$\xi(R,T) = \frac{1}{n} \cdot \sum_{i=1}^{n} \alpha(e_i, R, T, l_i)^2 \qquad (1)$$

where R and T are the rotation and translation components of the camera's rigid-body pose respectively, n is the number of correspondences, and α is the angle between the two planes spanned by the camera center and the observed image edge e_i and model segment l_i respectively.

4.1. Initialization

Initializing the camera pose requires determination of an initial set of valid correspondences between image edges and model segments. Rather than incur the geometric complexity of performing SFM from two or more images, then matching recovered to known 3D structure, we initialize from a single omnidirectional image. This simplifies the

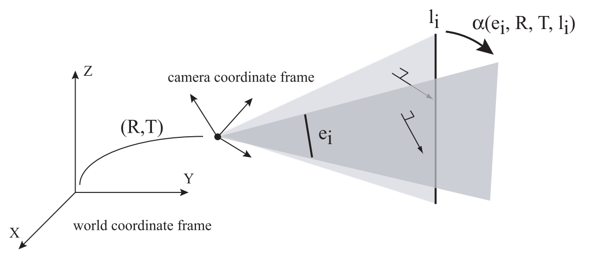

Figure 2. The angle α between image edge e_i and model segment l_i as seen by camera (R, T) is defined as the angle between the normals to the planes generated by the camera center and e_i and l_i respectively.

geometric computation required to a simple projection of model segments through our (known) camera model. The core idea of the initialization algorithm is to find the camera pose (R_0, T_0) that minimizes ξ. Neither exhaustive search of the 6-DOF space of poses, nor naive RANSAC [13], are tractable approaches when most model features are occluded. Instead, we search within a volume (center $\tilde{T}$, diameter δ) known to contain the camera position, and identify the 6-DOF camera pose within this volume that minimizes ξ. The time required for Initialization is proportional to the size of the search volume.

4.1.1 Model Coordinate Subdivision

The initialization algorithm uses a visibility data structure (described in § 4.3). This data structure consists of a discretization of the 3-DOF model space into *nodes* at a spacing of about one meter. Each node is associated with a volumetric *cell* containing all points closer to that node than to any other node (each cell is indeed the Voronoi region of one node, but the cell boundary is known by construction, rather than computed from the arrangement of node positions). The initialization algorithm is invoked with a specified search region. It identifies which cells intersect this region, and searches these cells for the camera pose with lowest ξ score.

4.1.2 Edge-Segment Matching

The core of the initialization algorithm is a method for generating and scoring putative correspondences between elements of two sets, m image edges $\mathcal{E}$ and n model segments $\mathcal{L}$, and using a selected subset of correspondences to recover camera pose. The method is based on the following observation: that a *pair* of image edges is a compatible match with a *pair* of model segments only if the dihedral angles formed by the two associated plane pairs differ by less than some bound determined by the inter-node distance. Using this observation, we define a function that takes as inputs a triplet of image edges and a triplet of model segments and returns a match score for the pairing of these triplets. The scoring function is defined as the normalized product of overlaps between the dihedral angle ranges taken over the cell interior. Given a set of image edges and a set of model segments, the algorithm computes the match score for each triplet of edges and segments and aggregates them within an $m \times n$ table.

The table also stores the k best candidate matches for each active model segment (we use $k = 3$). The initialization method next performs a series of random samplings pairing a model segment with one of its best-matching image edges. From each sample match set, the camera pose and resulting ξ value are computed; the algorithm returns the camera pose with the lowest ξ value. Figures 3 and 4 illustrate the algorithm.

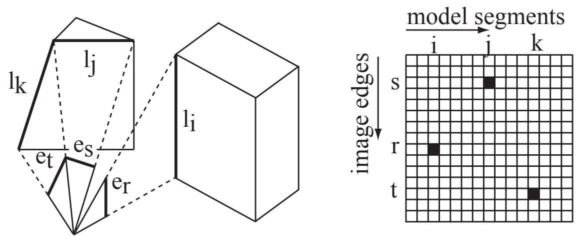

Figure 3. Edge-segment matching based on dihedral angle constraints. When a segment triplet $\{l_i, l_j, l_k\}$ is compatible with an edge triplet $\{e_r, e_s, e_t\}$, the scoring table is updated accordingly.

```
1:  Given $(\tilde{T}, \mathcal{E}, \mathcal{L}, \delta)$
2:  Initialize a $m \times n$ score table: $A = 0$
3:  **for** each triplet $\{l_i, l_j, l_k\} \in \mathcal{L}^3$ $(0 \le i < j < k < n)$
    and each triplet $\{e_r, e_s, e_t\} \in \mathcal{E}^3$ $(0 \le r < s < t < m)$
    **do**
4:    Compute min, max dihedral angles for $\{l_i, l_j, l_k\}$ and
      dihedral angles for $\{e_r, e_s, e_t\}$
5:    **if** dihedral angles match **then**
6:      Compute the overlap $\alpha$ between observed and expected dihedral angles
7:      Update: A[r][i]+= $\alpha$; A[s][j]+= $\alpha$; A[t][k]+= $\alpha$
8:  **for** each row $A^j$, $0 \le j < n$ **do**
9:    Determine top $k$ elements $\{A^j_{i_1}, \cdots, A^j_{i_k}\}$
10:   Generate multi-hypothesis correspondence
      $c_j = \{l_j \mid e_{i_1}, \cdots, e_{i_k}\}$
11: Draw random correspondence match set:
12: **for** each sample $\{c_{j_1}, \cdots, c_{j_p}\}$ **do**
13:   Select a random edge match for each $c_{j_k}$.
14:   Minimize the $\xi$ function over the sample.
15: Return solution $(R_0, T_0, \mathcal{S}_0)$ with lowest $\xi$ value.
```

Figure 4. The INIT-SEGMENT-EDGE Algorithm.

4.1.3 System Initialization

In practice, the system runs `INIT-SEGMENT-EDGE` given a coarse user bound around the camera position, and returns the solution with the lowest ξ value. For more robustness, the algorithm is followed by a standard simplex minimization of ξ. Figure 5 summarizes the Initialization phase.

```
1: Given $(\tilde{T}, \delta)$
2: Detect edges on the first frame $\rightarrow \mathcal{E}$
3: Determine the set of visible model segments $\mathcal{L} = \text{VIS}(T)$
4: Run INIT-SEGMENT-EDGE on $(\tilde{T}, \mathcal{E}, \mathcal{L}, \delta)$.
5: Keep the solution with the lowest $\xi(R, T)$ value.
6: Return the corresponding solution $(R_0, T_0, \mathcal{S}_0)$.
```

Figure 5. Initialization.

4.2. Maintenance

This section describes the algorithm's maintenance component. Given a set of edge-segment correspondences $\mathcal{S}_t$ at frame t, the maintenance problem is to identify a set of correspondences $\mathcal{S}_{t+1}$ at frame $t + 1$ and to compute the new camera pose (R_{t+1}, T_{t+1}). To account for clutter, we use a multi-hypothesis approach combined with a color-based inter-frame constraint.

4.2.1 Hue-based Edge Matching Constraint

Each image edge is associated with a hue mean and variance for two five-pixel wide regions, one on each side of the edge. Given a correspondence between an image edge and a model segment in frame t, the algorithm looks for matching edges in frame $t+1$ by considering all edges with a dihedral angle smaller than a given threshold (we use 10 degrees) and a hue distance smaller than a given threshold (we use 0.03 in a wrapped hue space $[0..1]$). We mark a correspondence *observed* if it satisfies these two criteria.

4.2.2 Motion Smoothness Assumption

Our system relies on the motion smoothness assumption in two ways. First, it uses a local visibility computation to determine the expected model segments in the next frame, given the camera position at the current frame (see § 4.3). Second, the hue-based filter described in § 4.2.1 incorporates a maximum angle threshold between two consecutive observations of a model segment on the image. Given a correspondence between an image edge and a model segment at frame t, only those image edges that fall within the angle threshold at frame $t + 1$ are candidates for the correspondence update (Figure 6).

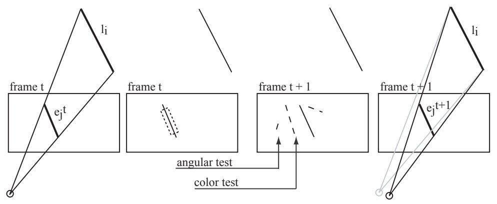

Figure 6. Correspondence update. Given a correspondence between the 2D image edge e_j^t and the 3D model segment l_i at frame t, the angular and hue constraints determine the most likely edge match e_j^{t+1} at frame $t + 1$.

4.2.3 Correspondence Subsets

After correspondence update, the system generates a series of random correspondence subsets. For each subset, the algorithm refines the camera pose (using the position at frame t for the initial guess), finds the inlier matches, refines again and finally computes the ξ function over the set of remaining correspondences. The pose with minimum ξ value is retained. This procedure tends to identify a consensus set of correspondences.

4.2.4 Correspondence Lifetimes

In order to further improve the robustness of the matching process, we implement a basic state machine for correspondences. The machine has three states: an entry state *unknown*, and two subsequent states *pending* and *accepted*. A correspondence status evolves to *pending* once it is observed, and to *accepted* if it is consistently observed over k consecutive frames (we use $k = 4$). A correspondence status degrades from *accepted* to *pending* if it is unobserved for at least one but no more than $k - 1$ frames, after which it either evolves to *accepted* or (when the edge has been unobserved in k consecutive frames) degrades to *unknown*. The color signature of an edge is retained as long as it is *accepted* or *pending*. However, only correspondences with *accepted* status are used for egomotion estimation. Figure 7 summarizes the Maintenance Algorithm.

The number of correspondences per sample λ is defined by the minimum number of correct correspondences required to accurately determine the camera pose. In theory, three correspondences are sufficient (omitting degenerate configurations). In practice, approximately 10 correspondences are needed to account for image and model noise (see § 5.1). The number of samples s_t is defined as the minimum number of draws of p out of q elements required to achieve 95% odds of success assuming the set has $b\%$ outliers, *i.e*. the minimum p such that :

$$(1 - \binom{(1 - \frac{b}{100})q}{p} / \binom{q}{p})^n \leq 5\% \qquad (2)$$

Table 1 evaluates s_t for different values of b and $|\bar{\mathcal{S}}_t|$.

	$|\bar{\mathcal{S}}_t| = 30$	40	50	60
$b = 10\%$	10	9	9	8
$b = 30\%$	254	193	167	152
$b = 50\%$	29971	13743	9413	7516

Table 1. Number of samples versus clutter percentage (b) and number of correspondences ($|\bar{\mathcal{S}}_t|$).

```
1:  Given $(R_t, T_t, \mathcal{S}_t)$
2:  Detect edges at frame $t + 1 \rightarrow \mathcal{E}$
3:  **for** each correspondence $\{l_i, e_j\}$ in $\mathcal{S}_t$ **do**
4:      Search for match in $\mathcal{E}$ satisfying color consistency
        with $e_j$ and acceptable angular error with $l_i$.
5:      If a match is found, update the correspondence.
6:  From the set $\bar{\mathcal{S}}_t$ of correspondences with *accepted* status, draw $s_t$ samples of $\lambda$ correspondences ($\lambda \ll |\bar{\mathcal{S}}_t|$).
7:  **for** each sample **do**
8:      Compute camera pose by minimizing $\xi$ over the $\lambda$
        correspondences using a simplex method.
9:      Score sample by computing $\xi$ over the remaining correspondences in $\bar{\mathcal{S}}_t$.
10: Keep sample with lowest score; update camera pose at frame $t + 1$.
11: Update each correspondence in $\mathcal{S}_t$ according to the state machine $\rightarrow \mathcal{S}_{t+1}$.
12: Query new visibility set $\mathcal{L}_{t+1} = \text{VIS}(T_{t+1})$
13: If $\mathcal{L}_{t+1} \neq \mathcal{L}_t$, remove demoted segments and insert new segments with status *unknown* in $\mathcal{S}_{t+1}$.
```

Figure 7. The Maintenance Algorithm

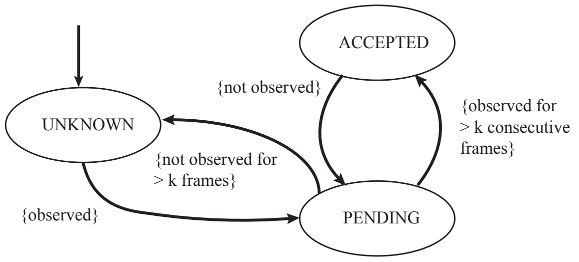
ACCEPTED
{not observed}
{observed for > k consecutive frames}
UNKNOWN
{not observed for > k frames}
{observed}
PENDING

Figure 8. Correspondence state machine.

4.3. Prior Visibility Analysis

In order for the system to scale well, an off-line process computes the set of visible model segments from each node in the 3D model. The data is stored in a lookup table and is queried during both Initialization and Maintenance, greatly reducing the number of candidate model segments that must be processed. The resulting look-up table accepts as input a 3D position T in the model and returns an estimate of the set of visible faces, segments and vertices $(\mathcal{F}, \mathcal{L}, \mathcal{V})$:

$$(\mathcal{F}, \mathcal{L}, \mathcal{V}) = \text{VIS}(T) \tag{3}$$

The visibility analysis is not conservative in a sense that it does not generate a superset of the edges visible to any point in the cell. However, given a sufficiently fine-grained sampling of the view space, even an underestimated visibility set will contain most prominent environment edges observed by the camera.

We use an OpenGL algorithm based on powerful modern GPUs (Graphics Processing Units) to perform the visibility analysis efficiently. First, the set of visible faces is computed by rendering the 3D model with a virtual camera centered at each node position, with each model face assigned a unique color. The contents of the framebuffer then correspond to the set of faces visible (at pixel resolution) from the node.

Second, we consider the set of model segments bounding at least one visible face. Each associated model segment is rendered through both the OpenGL feedback buffer and the OpenGL depth buffer (Figure 9). The feedback buffer contains the depth value at each pixel along the segment if the segment were to be displayed alone. The depth buffer contains the depth value at each pixel along the segment when rendering the full model. If the depth buffer value is equal to the feedback buffer value for at least two pixels along the segment (up to framebuffer precision), the segment is classified as visible. On the other hand, if the depth values differ at all pixels along the segment, the segment is classified as occluded by some model polygon. Figure 9 illustrates our algorithm.

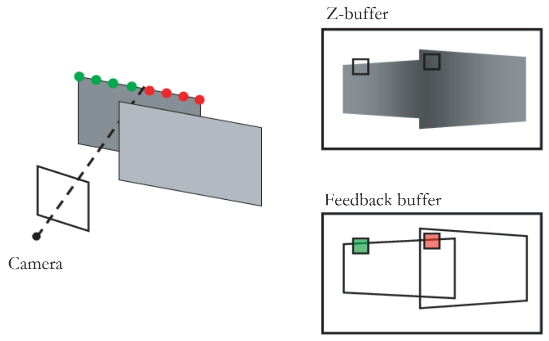

Figure 9. Visible segment determination using OpenGL and GPU.

5. Results

We demonstrate our system on one synthetic and three real image sequences:

- SYNTHETIC: 6-DOF motion within a simulated lab space;

- LAB: rolling 3-DOF (x, y, θ) motion within a real lab space;
- CORRIDOR: rolling 3-DOF motion through adjoining buildings; and
- HAND-HELD: hand-held 6-DOF motion within a real lab space.

Table 2 summarizes various attributes of the test sequences.

	lab	corridor	hand-held
Number of frames	1,500	7,800	1,900
Frame rate (s^{-1})	5	5	15
Excursion duration (min)	5	26	2
Excursion length (m)	120	936	33
Total # of 3D segments	3,000	7,400	3,000
Total surface area (m^2)	450	7,000	450

Table 2. Test sequence information.

Figure 1 shows the recovered camera motion for the LAB sequence. The initial camera pose was computed automatically using our initialization algorithm. Figure 12 shows an omnidirectional image and the re-projected 3D structure overlaid in green; note the high level of clutter and occlusion. Figure 10 shows the recovered motion for the CORRIDOR sequence. The 3D model was generated automatically from publically available 2D blueprints using a standard height for ceilings and door frames. Figure 11 shows the recovered egomotion for the HAND-HELD sequence. Figure 13 shows a detail view of long-range correspondence tracking over several hundred frames. Figure 14 shows the correspondence state for a 3D model segment being occluded during the sequence.

5.1. Localization Accuracy

Figure 15 shows the localization error (translation and rotation) for the SYNTHETIC sequence. We simulate image noise by convolving image edges with gaussian noise ($\sigma = .5\,\text{deg}$), and simulate clutter by removing 25% of the correspondences and adding gaussian noise of ($\sigma = 2\,\text{deg}$) to the remaining correspondences. Figure 16 shows the error in recovered position and orientation with respect to ground truth. The ground truth for position was obtained using a flat 2D motion for the camera and comparing the position component computed by the system along the z axis with the actual camera height measured with a laser range finder. The ground truth for rotation was obtained using an Xsens MTi gyro attached to the camera. The standard deviation is about 13 cm in position and two degrees in orientation.

5.2. Camera Calibration

The system uses the PointGrey Research Ladybug camera. The camera is composed of six CCD sensors covering more than 75% of the view sphere. The rigid-body transformation between each sensor and the virtual camera frame is provided by PointGrey Research. Due to their small focal length, the sensors are subject to high distortion. We calibrate each sensor independently using an ellipsoidal lens model [2].

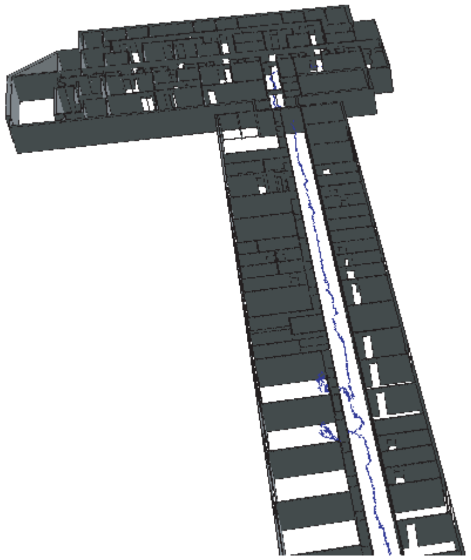
Figure 10. Recovered egomotion for CORRIDOR sequence (7,800 frames). Note motion into and out of adjoining offices. The 3D model was generated automatically from 2D blueprints and extrusion heights.

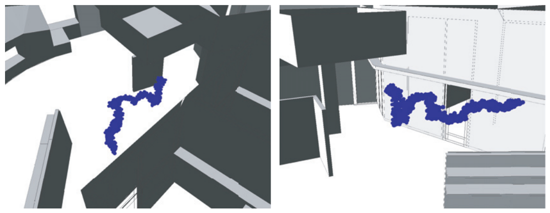
Figure 11. Recovered egomotion for HAND-HELD sequence (1,900 frames). Our system handles truly 6-DOF camera motion.

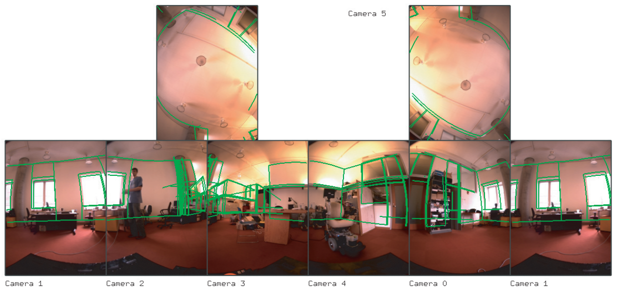

Figure 12. Omnidirectional image and re-projected 3D structure (in green). Localization is fairly accurate despite severe clutter. A video of this sequence is provided as supplemental material.

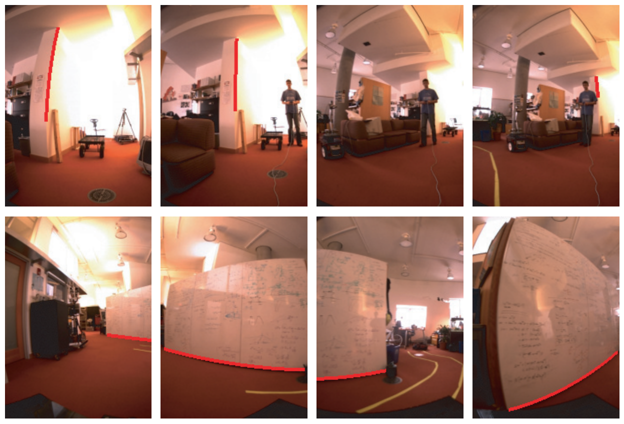

Figure 13. Top and bottom row: long-range correspondence tracking over several hundred frames (LAB sequence). Note recovery after occlusion (top row).

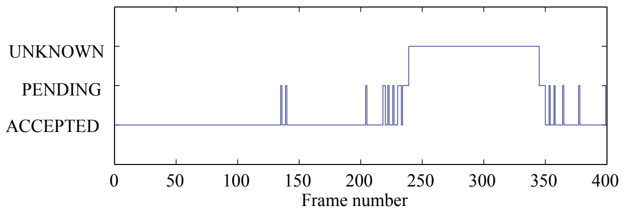

Figure 14. Correspondence state for a 3D segment being occluded from frame 240 to 350. The algorithm automatically detects the start and end of occlusion and maintains the correspondence state accordingly.

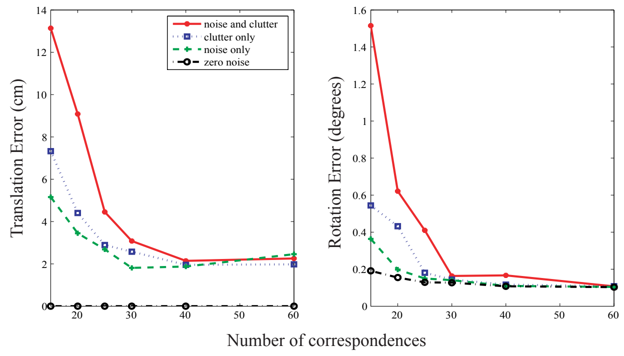

Figure 15. Localization accuracy with respect to the number of correspondences (simulated data, Gaussian noise on image edges $\sigma = 2\,\text{deg}$). Accuracy plateaus at about 40 correspondences.

5.3. Initialization

Figure 12 shows the result of Initialization on a real image using a search volume three meters wide. Figure 17 shows a correspondence generated by `INIT-SEGMENT-EDGE`. The model segment is shown in red. The three putative correspondences are shown on the image in blue and white. The algorithm succeeds in finding the correct match despite the high level of clutter.

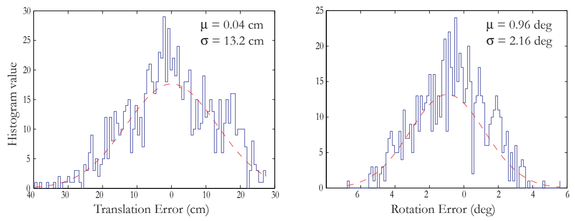

Figure 16. Position and orientation accuracy with respect to ground truth. (Gaussian fit shown as red dashed line).

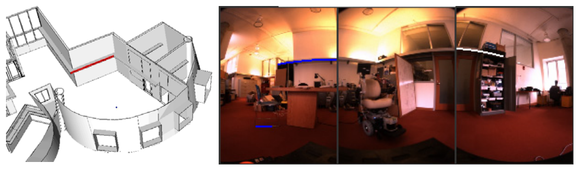

Figure 17. Putative correspondence determined by `INIT-SEGMENT-EDGE`. The algorithm determines the highest-scoring matches for the model segment (in red on the left) among the observed image edges. The three best candidates are displayed on the right. The correct match appears in blue on the left-hand side.

5.4. System Performance

The system currently runs at about 1Hz on a desktop PC with four 2GHz CPUs. Two thirds of the processing time are spent in edge detection and color processing (six 512×384, 8-bit images). The remaining time is spent in the random sample algorithm. The initialization phase takes one minute given a three-meter search volume. We have implemented an optimization for the special case of vertical camera pose which runs in about 10 seconds.

6. Discussion

This section discusses several limitations of the current method, and possible directions for its future development.

6.1. System Limitations

The system suffers from the following limitations. First, the method's performance could be improved, through more focused sampling, through code optimization, or with faster hardware. Second, the system's localization accuracy could be higher. Some error is surely due to feature localization; another error source is inaccuracy in the input 3D model. Third, the present sensor is not light-sensitive enough; it requires slow, smooth motion in order to avoid motion blur in indoor environments. Fourth, the initialization method can give ambiguous results in the presence of repeated environment structures such as multiple doorways along extended corridors. Finally, the visibility analysis assumes that a one-meter grid is fine enough to capture most variations in visibility.

6.2. Future Directions

We are currently pursuing several promising directions. First, a geometric signature-based initialization could enable the method to quickly eliminate inconsistent locations and cut down the number of regions in which to run the initialization algorithm. Second, integration of an inertial sensor and a camera motion model could increase the robustness of the maintenance phase. Third, we will investigate tracking of 3D points in addition to 3D segments. Finally, an on-line update of the model combined with occlusion processing could further decrease the occurrence of false matches.

7. Conclusion

We described an algorithm for 6-DOF localization from a coarse 3D model and an omnidirectional video sequence, based on establishing and maintaining matches between image edges and model segments. Our system makes few assumptions about the environment other than that it contains prominent straight line segments. Our solution algorithm employed two phases, initialization and maintenance, along with prior visibility analysis to drastically decrease running time and increase the scale of environments that can be handled by the method. We demonstrated the system, and evaluated its performance, on a variety of long-duration, spatially extended, visually cluttered image sequences.

Acknowledgements

We gratefully acknowledge the support of Draper Laboratory's University Independent Research and Development Program.

References

[1] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. In *Proc. ECCV*, volume 4, pages 282–296, New York, May 2002.

[2] M. Antone. An ellipsoidal lens model for fisheye calibration. *Technical Report TR-1940, BAE Systems Advanced Information Technologies, Computer Vision Group*, Nov. 2005.

[3] M. Antone and S. Teller. Scalable extrinsic calibration of omnidirectional image networks. *IJCV*, 49(2-3):143–174, 2002.

[4] A. Bartoli, R. Hartley, and F. Kahl. Motion from 3D line correspondences: Linear and non-linear solutions. In *Proc. IEEE CVPR*, pages 477–484, Madison, WI, June 2003.

[5] A. Bartoli and P. Sturm. Structure from motion using lines: Representation, triangulation and bundle adjustment. *CVIU*, 100(3):416–441, Dec. 2005.

[6] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas framework for scalable mapping. In *Proc. ICRA*, pages 1899–1906, Taipei, Taiwan, 2003.

[7] A. Clerentin, L. Delahoche, C. Pegard, and E. B. Gracsy. A localization method based on two omnidirectional perception systems cooperation. *Proc. ICRA*, 2:1219–1224, April 2000.

[8] S. Coorg and S. Teller. Matching and pose refinement with camera pose estimates. Technical Report MIT/LCS/TM-561, MIT, 1996.

[9] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1403, Washington, DC, USA, 2003. IEEE Computer Society.

[10] M. Dhome, M. Richetin, and J.-T. Lapreste. Determination of the attitude of 3D objects from a single perspective view. *IEEE Trans. PAMI*, 11(12):1265–1278, 1989.

[11] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Trans. PAMI*, 24(7):932–946, 2002.

[12] O. Faugeras, Q.-T. Luong, and T. Papadopoulou. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press, Cambridge, MA, USA, 2001.

[13] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[14] V. Gouet and B. Lameyre. SAP: A robust approach to track objects in video streams with snakes and points. *British Machine Vision Conference (BMVC'04)*, 2004.

[15] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision (2^{nd} ed.)*. Cambridge University Press, ISBN: 0521540518, 2004.

[16] B. K. P. Horn. *Robot vision*. MIT Press, Cambridge, MA, USA, 1986.

[17] G. Jang, S. Kim, J. Kim, and I. Kweon. Metric localization using a single artificial landmark for indoor mobile robots. *IEEE Intl. Conf. on Intelligent Robots and Systems*, pages 2857–2862, August 2005.

[18] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2D-3D model-based approach. In *Proc. IEEE ICCV*, pages 262–268, Sep. 1999.

[19] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777, 2004.

[20] L. Paletta, S. Frintrop, and J. Hertzberg. Robust localization using context in omnidirectional imaging. *Proc. IEEE ICRA*, 2:2072–2077, May 2001.

[21] L. Quan and Z.-D. Lan. Linear n-point camera pose determination. *IEEE Trans. PAMI*, 21(8):774–780, 1999.

[22] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE ICCV*, volume 2, pages 1508–1515, Beijing, China, 2005.

[23] R. Sim, P. Elinas, M. Griffin, A. Shyr, and J. Little. Design and analysis of a framework for real-time vision-based SLAM using Rao-Blackwellised particle filters. In *Proc. CRV*, pages 21–21, 2006.

[24] C. J. Taylor and D. J. Kriegman. Structure and motion from line segments in multiple images. *IEEE Trans. PAMI*, 17(11):1021–1032, 1995.