Monocular SLAM as a Graph of Coalesced Observations

Ethan Eade
ee231@cam.ac.uk

Tom Drummond
twd20@cam.ac.uk

Abstract

We present a monocular SLAM system that avoids inconsistency by coalescing observations into independent local coordinate frames, building a graph of the local frames, and optimizing the resulting graph. We choose coordinates that minimize the nonlinearity of the updates in the nodes, and suggest a heuristic measure of such nonlinearity, using it to guide our traversal of the graph. The system operates in real-time on sequences with several hundreds of landmarks while performing global graph optimization, yielding accurate and nearly consistent estimation relative to offline bundle adjustment, and considerably better consistency than EKF SLAM and FastSLAM.

1. Introduction

Simultaneous localization and mapping (SLAM) with a single camera is an especially interesting and difficult form of the general problem. Solutions attempt to estimate environment structure and camera trajectory online, under a highly nonlinear partial observation model. Over recent years, the vision community has explored several approaches to the problem. These systems use results and methods from structure-and-motion work, and from the robotics literature, where SLAM has been a busy topic for some time. Our work is motivated by both perspectives.

Recursive or causal estimation, in which the state estimate of the system depends only on observations up to the current time, is a common element of SLAM solutions, which must operate online. The extended Kalman filter (EKF) provides the recursive estimation machinery for several vision SLAM systems[1, 7, 11, 8]. Davison[8] successfully employs an EKF to perform real-time estimation in monocular SLAM. The EKF linearizes the observation and dynamics models, yielding a Gaussian posterior on state. This representation is faithful to the true posterior only when the models are nearly linear. In the case of perspective projection of Cartesian points with a single camera, the observation model is far from linear.

Davison addresses this problem pragmatically by using a two-stage initialization process involving a particle filter. More recent approaches[9, 16] show that the observation model becomes nearly linear (for small camera displacements) when using an inverse depth representation.

The monocular SLAM system of [9] uses the FastSLAM algorithm[15] to perform recursive estimation with a Rao-Blackwellized particle filter. The focus of the work is on real-time operation with many landmarks, which is especially difficult with the EKF due to its computational complexity.

However, for the nonlinear observation models of visual SLAM, both the EKF and FastSLAM approaches are inconsistent[3, 4]. A consistent filter generates a posterior estimate on state without synthesizing information not present in observations. Inconsistency implies underestimation of uncertainty, which prevents convergence to the correct solution and can lead to catastrophic failure. In the EKF, inconsistency results from imperfect approximation of the observation model by a linearization, and representation of uncertainties in a common global frame. In FastSLAM, the limited number of particles used to represent map uncertainty also causes inconsistency.

Standard offline global bundle adjustment avoids inconsistent posteriors, as each iteration of the nonlinear optimization re-linearizes the model around the current state for all observations[21]. Errors in linearization do not propagate from one iteration to the next. Unfortunately, standard bundle adjustment algorithms cannot be feasibly employed for on-line SLAM, as they require storage of all observations, and the computational requirements of optimization grow rapidly with the number of observations and landmarks.

Some basic approaches to hierarchical bundle adjustment have been explored. The method of [10] first estimates trifocal tensors over view triplets, then later performs a full bundle adjustment over all these subsets. However, the aim is not to make the bundle adjustment more efficient, but instead to get a starting point for bundle adjustment without performing SLAM. The system is not intended for online use (and such a target is unattainable with a full bundle adjustment stage).

The system described in [18] and [17] (both papers describe the same system) performs bundle adjustment over

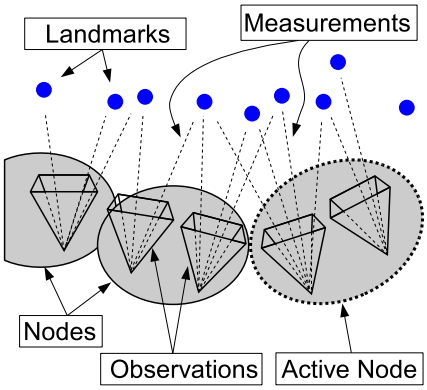

Figure 1. Observations are coalesced in nodes (local coordinate frames.

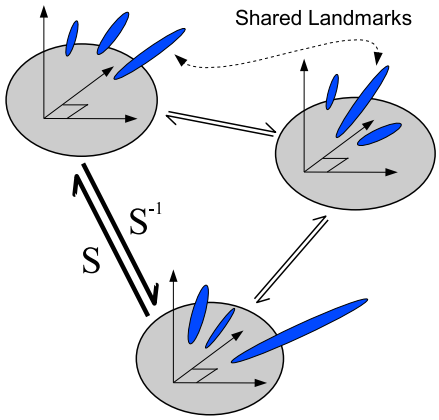

Figure 2. Common estimated landmarks in nodes induce similarity transformations between the nodes.

a constant number of recent key frames, which are chosen sparsely from the set of all views using simple heuristics. While performing such temporally local bundle adjustment improves the accuracy of the result, for sufficiently large sequences it will not converge to the solution given by a full optimization. It offers a quantitative, instead of a qualitative, improvement over basic SLAM. Statistical consistency is not evaluated.

Our approach can be viewed as a hierarchical bundle adjustment algorithm, in which multiple observations sharing a nearly-linear observation model are coalesced into nodes containing high-dimensional, rich observations, and the relations between these high-dimensional observations are optimized at the global level. Thus optimization of the linear parts of the parameter space proceeds recursively, permitting global optimization at orders-of-magnitude less cost than bundle adjustment. No video frames in the sequence need be privileged as key frames. No observations are ignored or forgotten; all are naturally assimilated into the filter and constrain the optimization.

We propose a monocular SLAM algorithm that employs recursive estimation in local coordinate frames with a nearly linear observation model, but performs further optimization iteratively over a graph of such frames. The system avoids inconsistency by not propagating linearization errors through the global posterior, and by always representing uncertainties locally. Several SLAM systems in the robotics literature represent state using multiple local coordinate frames [6, 13, 14, 5, 2]. The Atlas[5] framework and H-SLAM[14] build graphs, but do not optimize cycles in the graph. The Network Coupled Feature Maps (NCFM) approach[2] performs optimization over graph cycles. Of these approaches, our system most resembles Atlas and NCFM.

We make the following contributions in this work: We describe, to our knowledge, the first implementation of real-time monocular SLAM using multiple local coordinate frames. We avoid inconsistency in our SLAM estimate by using local coordinates that make the observation model nearly linear. Furthermore, we quantify the nonlinearity of the model, allowing us to choose the optimal local coordinate frame to update with new observations. The system uses top-down active search, yielding reliable data association and allowing efficient operation. We show results of the system on indoor sequences, and compare its estimates to EKF SLAM and FastSLAM, as well as to bundle adjustment.

2. Representation of State

We represent our state of knowledge about the world as a graph. The nodes of the graph contain information from distinct sets of observations, with an observation defined as a set of landmark measurements in a single video image. The nodes are constrained such that these observations can be combined using a (nearly) linear observation model, providing an estimate of landmark positions in a local coordinate frame. The edges of the graph represent the transformations between the coordinate frames of nodes that contain estimates of common landmarks. Because the scale within each node is a free parameter, the transformations must represent scale changes between nodes. Thus the edge transformations are scaled Euclidean transformations – i.e., similarity transformations.

2.1. Similarity Transformations

A similarity transformation S is given by a rotation and translation $(R, T) \in$ SE(3) and a scale s: $S = [(R, T), s]$. A Cartesian point $\mathbf{x}$ is mapped via S as

$$S(\mathbf{x}) \equiv s\,(R\mathbf{x} + T) \tag{1}$$

Analogously to the rigid transformation group SE(3), differentials and covariances of similarity transforms are represented in the tangent space around the origin, and mapped into the space via the exponential map. A camera pose (R_C, T_C) maps through $S = [(R, T), s]$ by

$$(R_C, T_C) \overset{S}{\mapsto} \left(R_C R^T, s \cdot (T_C - R_C R^T T)\right) \quad (2)$$

2.2. Coalescing Observations in Nodes

Each node maintains Gaussian estimates of landmarks observed locally. For the filter to be consistent, the actual posterior distributions of the landmark coordinates must be well represented as Gaussians. However, a perspective projection model with Gaussian observation noise yields highly non-Gaussian posteriors in Cartesian coordinates: Two observations of a landmark with a narrow baseline give a cone-shaped distribution in space.

The Gaussian observations fail to induce a Gaussian posterior because the observation model $\mathbf{f}(\mathbf{x})$ is considerably nonlinear: the projection to the image plane involves division by the depth of the landmark. Consider the projection function for a camera pose in the local coordinate frame with translation $\mathbf{T}$ and no rotation:

$$\mathbf{f}(\mathbf{x}) = \pi(\mathbf{x} + \mathbf{T}) \quad (3)$$

$$= \frac{1}{z + \mathbf{T}_3} \begin{pmatrix} x + \mathbf{T}_1 & y + \mathbf{T}_2 \end{pmatrix} \quad (4)$$

Even for $\mathbf{T}_3 = 0$, the projection is nonlinear in the landmark coordinates. We choose instead an inverse depth representation for landmarks, given in terms of local Cartesian coordinates by

$$\begin{pmatrix} u & v & q \end{pmatrix}^T \equiv \frac{1}{z} \begin{pmatrix} x & y & 1 \end{pmatrix}^T \quad (5)$$

The observation model is now nearly linear in landmark coordinates:

$$\mathbf{f}\left(\begin{pmatrix} u & v & q \end{pmatrix}^T\right) = \pi\left(\left(\frac{1}{q}\begin{pmatrix} u & v & 1 \end{pmatrix}^T + \mathbf{T}\right)\right) \quad (6)$$

$$= \pi\left(\begin{pmatrix} u & v & 1 \end{pmatrix}^T + q\mathbf{T}\right) \quad (7)$$

$$= \frac{1}{1 + q\mathbf{T}_3} \begin{pmatrix} u + \mathbf{T}_1 & v + \mathbf{T}_2 \end{pmatrix}^T \quad (8)$$

When the local depth of the landmark is large relative to the displacement of the camera from the origin, we have $q\mathbf{T}_3 << 1$, and the nonlinearity vanishes. The observation function remains linear for any camera translation in the X-Y plane, and for any rotation purely around the Z axis of the coordinate frame. We quantify the nonlinearity using the Hessian (see Sec. 3.2). We represent the state of all landmarks in a node with a mean vector (in inverse depth representation) , and a full information matrix (inverse covariance) on the coordinates.

3. SLAM Algorithm

The framework described in Sec. 2 could be employed in offline structure and motion estimation, performing bundle adjustment with substantially reduct computational cost. However, SLAM requires more than this, and we describe its operation in this section.

A single calibrated camera provides video images to the system. At any time, one node is designated *active*. Each time a video image is retrieved from the camera, the following takes place:

1. **Active Search:** Landmark locations are predicted using existing state estimates in and around the active node. Constrained searches for predicted landmarks in the image yield a set of noisy measurements (an observation).

2. **Choosing the active node:** The node where the observation model is most linear will best accommodate the observation. A new node might be created if no suitable one exists. The chosen node becomes active.

3. **Updating the local state:** The active node incorporates the observation into its local state estimate, avoiding inconsistency in the computation.

4. **Updating the graph:** Nodes with estimates of common landmarks are connected with edges. The node states and any cycles in the graph provide constraints on graph edges, which are optimized globally.

3.1. Active Search

We perform constrained searches in the video image to measure landmarks. A node associates a small image patch with each landmark it has seen, taken from the first local measurement of the landmark. The active node's landmark estimates are projected into the image through the observation model, and the gated uncertainty ellipses describe the region of the image where the landmark should appear with high confidence.

We scale and rotate the landmark's patch according to the current estimate of camera pose, relative to the pose in which the patch was acquired. This accounts, to first order, for appearance changes of the patch due to camera motion. The warped patch is sought in a gated region using normalized cross correlation. All correlations above a threshold contribute to the measurement estimate (location and noise). Measurements with a high spatial noise in the image are discarded, avoiding spurious measurements of repeated structure.

We do not base our predictive search solely on information in the active node; we also incorporate estimates from nearby nodes in the graph. A distance-limited breadth-first search of the node starting from the active node yields a tree

connecting nearby nodes. We propagate a landmark's estimates through this tree into the active node (the root). This propagation progresses in a depth-first manner, with each node first collecting estimates from its children in the tree before mapping the combination through the edge to its parent. Thus the uncertainty of each coordinate transformation is reflected properly in the result.

When the estimates reach the the active node, any local information about the landmark is added to the prediction. This yields a posterior estimate of the landmark in the local coordinate frame, conditioned on the information in all nodes of the tree. This posterior is projected through the observation model, yielding a refined search region in the image.

Thus, even if a node does not have a direct estimate of a landmark, or even if its estimate is very uncertain, by combining the information from nearby nodes, a feasible prediction can be made for the landmark. Note that the internal state of all nodes in the tree remains unchanged by this process; it serves only to aid the active search.

When the ratio of failed to successful searches for a landmark reaches a threshold, or when the search fails many times consecutively, the landmark is dropped from the active node. Nodes manage landmark memberships independently, so dropping a landmark from the active node does not cause it to be removed from other nodes. Thus, the set of local nodes maintaining estimates of a landmark L is effectively a visibility map and appearance model of L.

3.2. Choosing the Active Node

For a given observation (set of measurements) made from a video image, we choose to update one node's state estimate. This will be the active node. The process of coalescing observations into a node yields a Gaussian estimate of the posterior in the node's coordinate system. This representation is faithful only when the observation model is nearly linear. Otherwise, information is synthesized and the filter becomes inconsistent. The choice of inverse depth representation makes perspective projection nearly linear, compared to the highly nonlinear projection of Cartesian coordinates. However, the model will still lose linearity with sufficient camera motion away from the node's origin. We quantify the nonlinearity of the observation model as a function of the camera pose in order to choose the best node to update.

An observation model linear in the landmark parameters will have a Hessian of zero in those parameters. Thus the 'magnitude' of the Hessian gives a measure of the departure from linearity. We choose the Laplacian of the observation model with respect to the landmark (the trace of the Hessian) as a measure of the nonlinearity. That is, evaluating the Laplacian of the projection of inverse depth point $\mathbf{p} = \begin{pmatrix} u & v & q \end{pmatrix}$ through camera pose $C = (R, T)$ yields a vector that describes how much the Jacobian changes when $\mathbf{p}$ is perturbed. We use the magnitude of this vector as our measure:

$$\begin{pmatrix} u' & v' \end{pmatrix}^T \equiv \pi\left(C\left(\mathbf{p}\right)\right) \tag{9}$$

$$\mathbf{nl}(C, \mathbf{p}) \equiv \left| \left(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2} + \frac{\partial^2}{\partial q^2} \right) \begin{pmatrix} u' & v' \end{pmatrix}^T \right| \tag{10}$$

To get a heuristic estimate of nonlinearity for a pose without considering a specific landmark, we note that the scale of local node landmark estimates is normalized to be near unit depth. So we define our heuristic

$$\tilde{\mathbf{nl}}(C) \equiv \mathbf{nl}\left(C, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^T\right) \tag{11}$$

To accommodate uncertainty in the camera pose, we use the unscented transform[20] to map a Gaussian estimate of camera pose to a Gaussian estimate of nonlinearity.

We consider the current active node and all of its neighbors in the graph as candidates for the update. A candidate is viable only if it has estimates of at least k of the measured landmarks, and the nonlinearity of the camera pose (mapped into the candidate node) is below a predefined threshold m. Typical values are $k = 6$ and $m = 0.75$. The value k must be large enough to ensure that the state update in the node will be well-conditioned.

Assuming that multiple viable candidates exist in the local neighborhood, the candidate with the least nonlinearity wins. When no candidate is viable, a new node is created (see Sec. 3.6) and made active. Given a viable candidate, the camera pose and its uncertainty are projected through the edge to the chosen node, which now becomes the active node. The local state update is performed in the active node.

3.3. Local State Update

Given an observation – a set of two-dimensional Gaussian measurements of landmarks – updating a node's mean vector μ and information matrix $\mathbf{I}$ reduces to standard nonlinear optimization. Consider an observation $\mathbf{z}$ with (block diagonal) noise $\mathbf{R}$, and a camera pose starting point C. Let the observation model be given by $\mathbf{g}$. We wish to maximize the likelihood of $\mathbf{z}$ under our prior (μ and $\mathbf{I}$) by adjusting μ by $\delta\mu$ and the pose by δC. Equivalently, we minimize the negative log-likelihood E of the observation and the updates:

$$\mathbf{v} \equiv \mathbf{z} - \mathbf{g}(\mu + \delta\mu, C + \delta C) \tag{12}$$

$$E = \mathbf{v}^T \mathbf{R}^{-1} \mathbf{v} + \delta\mu^T \mathbf{I} \delta\mu \tag{13}$$

This system is identical to bundle adjustment with one view and a prior on structure given by $\mathbf{I}$. We reduce the system using the Schur complement, as described in detail in

[21], marginalizing the camera pose out of the result. We employ the Cholesky decomposition to perform Levenberg-Marquardt minimization of E. Note that the scale of structure in the local node is a gauge freedom – scaling all the landmarks' q coordinates does not affect E. We fix the gauge at each iteration by adding a large prior along the direction of current q coordinates (which keeps the optimization from moving the state uniformly along that vector). When the optimization is complete, we normalize the geometric mean of the inverse depths (and scale the corresponding elements of $\mathbf{I}$ by the same factor), again fixing the gauge.

Note that we impose no prior on pose C: The pose 'guess' serves merely to start the optimization in the right minimum. Adding any information about the current pose estimate to the optimization would make the result covariant with the pose. This would violate the statistical independence of distinct local nodes, as the pose estimate would carry information from one node's state into another's.

The iterative minimization yields $\delta\mu$ and an updated information matrix $\mathbf{I}'$, as well as a pose update δC with associated information ${\Sigma_C}^1$. We find that the minimization almost always (in more than 99% of updates) converges in one iteration, which reflects the near linearity of the observation model. The state update effectively coalesces the new observation into the local node's single merged high-dimensional observation. The update cost is cubic in the dimension of the node's state, but in our implementation, updates incorporating observations of 30 measurements into nodes with 50 landmarks consume a small fraction (less than 5 milliseconds) of our computation budget. By bounding the number of landmarks estimated in each node, we also bound the cost of updates.

3.4. Landmark Acquisition

When an insufficient number of known landmarks is observable in the current video image (due to movement into new territory), the system must acquire new landmarks to track. We use the FAST corner detector[19] to choose well-textured points in the image, and avoid regions around the projections of existing landmarks in the given image. The image patch surrounding the corner is stored so that the landmark may be observed in subsequent images (using the warping described above).

Initialization of new landmarks has received much attention in monocular SLAM[8, 12, 9, 16], as incorporating partially-observed state into a SLAM filter is tricky. However, because we use an inverse depth representation for landmarks, and because we store the state in an information form, partial initialization becomes trivial.

Consider a newly chosen landmark L, with a chosen location p in the image and unit-pixel-radius Gaussian noise in p. Mapping p and the noise through the calibrated camera model yields a point $z = (u, v)^T$ in the image plane of the current pose, with associated noise $\mathbf{R_z}$. This is the first (and only) measurement of L. We map the image plane point $(u, v, 1)$ through the current camera pose $C = (R, T)$ to get an inverse depth representation $\mathbf{p}$ in the local frame:

$$\mathbf{X} \equiv R^T \begin{pmatrix} u & v & 1 \end{pmatrix}^T - R^T T \tag{14}$$

$$\mathbf{p} = \frac{1}{\mathbf{X}_3} \begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 & 1 \end{pmatrix}^T \tag{15}$$

The mean of L (in μ) is initialized to $\mathbf{p}$ to provide a reasonable linearization point for the iterative optimization. The rows and columns of $\mathbf{I}$ corresponding to L are zeros, reflecting that we have not yet incorporated any information about L. The measurement $(\mathbf{z}, \mathbf{R_z})$ is added to the update along with any other measurements made in the latest image. The update optimization then yields the correct posterior on L in the local node's coordinates, which has information in only two of three dimensions. Crucially, it also correctly fills in the cross-information between L and other landmarks measured in the same video image. Subsequent measurements of L will yield estimates of its yet-unobserved dimension (the depth of L in the camera frame of C).

3.5. Updating the Graph

For any two nodes sharing a sufficient set of landmarks, a similarity transformation between the nodes is implied by the estimates of common landmarks. This similarity transformation is made explicit by estimating it and storing it in an edge linking the two nodes in a graph. An edge encodes the data-association information that links estimates in one local node to those in another. Edges are considered undirected, but the transformation along one direction is the inverse of that along the other direction.

An instantiated edge maintains both the (7-dimensional) Gaussian transformation estimate (S_L, Σ_L) induced locally by the two endpoints and a separate similarity transformation S_G that is modified by the graph optimization algorithm, taking into account (S_L, Σ_L) (see Sec. 4). Point and camera estimates are projected through the edge using (S_G, Σ_L), which gives the optimized mean and a conservative (over) estimate of covariance Σ_L.

The estimate (S_L, Σ_L) is computed from the shared landmark estimates in the endpoints. Iterated Gauss-Newton optimization modifies the parameters of S_L to maximize the likelihood of landmark estimates in the source projected through (S_L, Σ_L) into the target. The more precise the local estimates of landmarks in the source and target, the more precise the resulting transformation estimate. In contrast to the algorithm of Atlas[5], whenever a local update is performed in one of the endpoint nodes, or when one of the endpoints modifies its local scale, the edge trans-

formation is recomputed to reflect the change. Further optimization of S_G is described below in Sec. 4.

3.6. Creating Nodes

When no existing node satisfies the requirements given in Sec. 3.2, a new node is created. An edge is created from the current active node to the new node. The local transformation estimate S_L of the edge is initialized to the pose estimate in the active node, with unit scale. The new node becomes active, with the camera pose mapping to the identity and the pose uncertainty vanishing. The observation from the current video image is added directly to the new node's local state, without performing an update optimization.

Until the new node has undergone several local updates, the transformation estimate is not recomputed, as it will be poorly constrained. Once the new node has acquired a non-degenerate local structure estimate, the transformation estimate is computed and maintained.

3.7. Creating Edges

After the active node performs a local update, the neighborhood of the graph around the active node is explored with a breadth first search. If a node within a fixed (graph) distance of the active node shares a sufficient number of landmarks with the active node, a new edge is created between them. In our implementation, we require 10 shared landmarks. The local transformation estimate of the edge is immediately computed from the common landmark estimates.

4. Graph Optimization

At each time step, after the observation has been processed and new nodes and edges created, we perform global optimization on the current graph. The optimization modifies only the transformation parameters S_G in the graph edges, leaving the coalesced node observations untouched (and independent).

Two types of constraints are at work in the graph. First, shared landmarks between nodes induce local constraints encoded in each edge by (S_L, Σ_L). Second, the rigid geometry of three-space implies that any edge path from a node back to itself should compose to the identity transformation. So each cycle in the graph yields a constraint.

Our optimization seeks the maximum likelihood graph in which cycles compose to the identity. Consider edge transformations a, b, c, respectively joining nodes $A \to B$, $B \to C$, and $C \to A$. With juxtaposition implying composition, the cycle constraint implies

$$cba = 1 \tag{16}$$

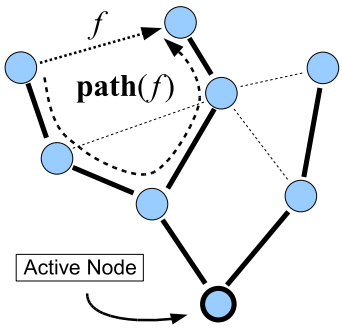

Figure 3. Cycle constraints.

So each cycle constraint removes an edge from the optimization, to be determined as a function of the remaining edges such that the cycle satisfies the constraint.

We detect cycles in the graph by building a spanning tree, T. Recall that a spanning tree touches all nodes in the graph. We build T by doing a full breadth first search of the graph. Each edge $f \notin T$ yields a cycle path, given by the edges $c_i \in T$ forming the path from one end of f to the other. Let this composed path in the tree be $\mathbf{path}(f)$. Let the local constraint (S_L, Σ_L) for an edge g be $\mathbf{local}_g$. With E the set of all edges in the graph, we define a likelihood:

$$L(T) = \prod_{e \in T} L(e|\mathbf{local}_e) \cdot \prod_{f \in E \setminus T} L(\mathbf{path}(f)|\mathbf{local}_f) \tag{17}$$

We minimize the negative log-likelihood using preconditioned gradient descent. We precondition by multiplying the standard gradient descent search direction by the inverse of the block-diagonal approximation of the Hessian. This is equivalent to Gauss-Newton optimization with a block-diagonal Hessian. We find that the preconditioning greatly speeds convergence.

Global graph optimization typically converges in fewer than 15 iterations each time step, requiring a small portion (20%) of the computation budget. Because the graph is optimized at each time step, it is never far from the minimum, except when long cycles appear for the first time, introducing new constraints on several edges. In these cases, the optimization converges rapidly over several time steps.

5. Results

Our system runs in real time on a standard 2.8 GHZ Pentium IV workstation. It processes 640x480 VGA frames captured from a USB2 camera at 30 Hz. The computation break-down is shown in Fig. 4. Examples of operation and output are shown in Fig. 8, Fig. 9 and Fig. 7.

Task	Time (ms)
Corner Detection	3
Landmark Prediction	2
Landmark Search	10
Local Update	5
Graph Optimize	4
Visual Rendering	4

Figure 4. Processing time per frame, while measuring 30 landmarks per frame in a graph with approximately 50 nodes, 200 edges, and 60 cycles. There are more than 400 landmarks estimated in the graph.

For the purposes of evaluation, we run our SLAM system on three indoor sequences of length roughly one minute each, recording all measurements as it runs. Then EKF SLAM, FastSLAM, and bundle adjustment are run on the recorded set of observations. Bundle adjustment converges to the same solution given the any of the SLAM solutions as a starting point.

We use the FastSLAM implementation described in [9]. We use an EKF SLAM implementation similar to that described in [8], trying both a Cartesian coordinate system for landmarks, and an inverse depth coordinate system. The consistency and accuracy when using inverse depth are superior for our sequences, so we present only the better EKF results for comparison. All the SLAM systems use a constant-velocity motion model with identical process noise.

Our bundle adjustment performs a Levenberg-Marquardt optimization over a reduced system, as described in detail in [21]. At each iteration, we normalize the map scale, to keep the gauge from drifting. The optimization ceases when the residual cannot be decreased by at least a factor of $1 - 10^{-12}$.

For each SLAM algorithm, at the end of processing we compute the covariance of all the landmarks projected into a common (world) frame, and compute the corresponding covariance over landmarks given by the bundle adjustment. Denote the SLAM posterior covariance by $\mathbf{P}_S$ and the covariance given by the iterative bundle adjustment by $\mathbf{P}_B$. A consistent estimate should yield a positive definite difference:

$$\mathbf{P}_D = \mathbf{P}_S - \mathbf{P}_B \tag{18}$$

Intuitively, a consistent estimate will converge to the true solution when given more and more information, thus shrinking dimensions of the covariance constrained by the additional measurements. We compute the difference of covariances and examine its eigen-decomposition. We also compute structure errors between the maps given by the filters and that given by bundle adjustment.

In our indoor sequences, our SLAM algorithm yields no worse than 5% negative eigenvalues. This implies that 5% of the state estimate dimensions have become inconsistent. The EKF and FastSLAM implementations run on the same sequences yield worse than 48% inconsistency (Fig.5).

Sequence	Local SLAM	FastSLAM	EKF
a	2%	55%	66%
b	5%	52%	48%
c	3%	80%	68%

Figure 5. Percent of eigenvalues of $\mathbf{P}_D$ which are negative: 0% is fully consistent.

Sequence	Local SLAM	FastSLAM	EKF
a	2.48	16.78	15.59
b	3.45	11.46	9.43
c	2.15	8.34	3.40

Figure 6. Reconstruction errors: the root mean residual of landmark positions compared to bundle adjustment, after registering the maps.

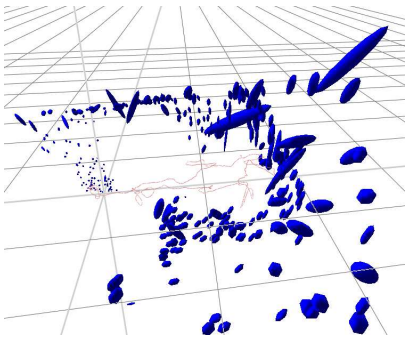

Figure 7. The map posterior at the end of a sequence. The map contains more than 400 landmarks. For display, all landmark estimates have been projected to a common coordinate frame through the graph edges, as described in Sec. 3.1. No offline optimization has been performed.

To compare the accuracy of the generated maps, we check the residual of the landmark positions compared to bundle adjustment (using the covariances given by the posteriors of bundle adjustment and SLAM). First the maps are aligned with the posterior of bundle adjustment with a similarity transformation optimized to minimize this residual. As shown in Fig.6, our SLAM method gives better reconstruction accuracy than either the EKF or FastSLAM on our sequences.

6. Conclusion

We have described a SLAM system that avoids inconsistent estimation by coalescing observations into local coordinate frames, represented in a graph. The system employs an inverse depth representation locally to minimize the nonlinearity of the perspective projection observation model.

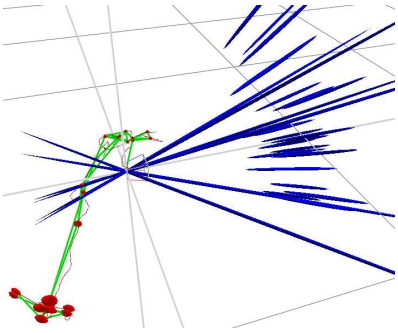

Figure 8. During SLAM operation. The graph nodes (red) are shown relative to the active node, and edges are drawn in green between them. The system is observing new landmarks.

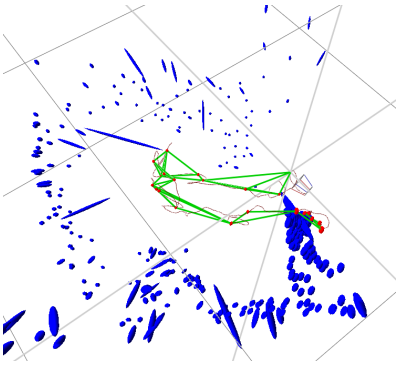

Figure 9. The end of a sequence. The graph contains 22 nodes, 36 edges, and 15 cycles. All Landmarks are projected through the graph into the active node for display purposes.

We have also shown how to estimate the nonlinearity of the model, so that the most appropriate graph node updates its estimate from observations each time step.

Our initial evaluation demonstrates that the system operates efficiently, producing posteriors for large maps (hundreds of landmarks) that are nearly consistent relative to bundle adjustment. Furthermore, global graph optimization over cycles is easily computed each time step within the bounds of computation.

References

[1] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *PAMI 1995*, 17(6):562–575, June 1995.

[2] T. Bailey. *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis, ACFR, Univ. of Sydney, August 2002.

[3] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the ekf-slam algorithm. In *IEEE/RSJ IROS 2006*, Beijing, China, October 2006.

[4] T. Bailey, J. Nieto, and E. Nebot. Consistency of the fastslam algorithm. In *IEEE ICRA 2006)*, Orlando, USA, May 2006.

[5] M. Bosse, P. Newman, J. Leonard, M.Soika, W. Felten, and S. Teller. An atlas framework for scalable mapping. In *ICRA*, pages 1899–1906, Taiwan, April 2003.

[6] J. A. Castellanos, R. Martinez-Cantin, J. D. Tardos, and J. Neira. Robocentric map joining: Improving the consistency of ekf-slam. *Robot. Auton. Syst.*, 55(1):21–29, 2007.

[7] Chiuso, Favaro, Jin, and Soatto. Structure from motion causally integrated over time. *IEEE PAMI 2002*, 24(4):523–535, April 2002.

[8] A. Davison. Real time simultaneous localisation and mapping with a single camera. In *ICCV*, Nice, France, July 2003.

[9] E. Eade and T. Drummond. Scalable monocular slam. *cvpr*, 1:469–476, 2006.

[10] A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *ECCV 1998*, pages 311–326, June 1998.

[11] H. Jin, P. Favaro, and S. Soatto. A semi-direct approach to structure from motion. *The Visual Computer*, 19(6):377–394, Oct 2003.

[12] T. Lemaire, S. Lacroix, and J. Sola. A practical 3d bearing-only slam algorithm. In *IROS 2005*, August 2005.

[13] J. J. Leonard and P. M. Newman. Consistent, convergent, and constant-time slam. In *IJCAI*, pages 1143–1150, 2003.

[14] Lisien, Morales, Silver, Kantor, Rekleitis, and Choset. Hierarchical simultaneous localization and mapping. *IROS 2003*, pages 448–453, October 2003.

[15] Montemerlo and Thrun. Simultaneous localization and mapping with unknown data association using fastslam. In *Proc. of IEEE ICRA 2003*, Taipei, 2003.

[16] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular slam. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[17] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. 3d reconstruction of complex structures with bundle adjustment: an incremental approach. In *ICRA 2006*, pages 3055–3061, Orlando, USA, May 2006. IEEE Computer Society.

[18] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Monocular vision based slam for mobile robots. In *ICPR 2006*, pages 1027–1031, Washington, DC, USA, 2006. IEEE Computer Society.

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

[20] J. K. U. S. J. Julier. A new extension of the kalman filter to nonlinear systems. In *The Proceedings of AeroSense*, pages 1628–1632, Orlando, Florida, USA, 1997. SPIE.

[21] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000.