Bag-of-Words-driven Single Camera Simultaneous Localisation and Mapping

Tom Botterill
Geospatial Research Centre
University of Canterbury
Christchurch 8041, NZ
tom.botterill @ grcnz.com

Steven Mills
Areograph Ltd.
90 Crawford St.
Dunedin, NZ

Richard Green
Department of Computer Science
University of Canterbury
Christchurch 8041, NZ

October 4, 2011

This is the pre-peer reviewed version of the following article: "Bag-of-Words-driven Single Camera SLAM", which is to be published in full in the Journal of Field Robotics (`http://www.journalfieldrobotics.org/`).

For additional details, please see my PhD thesis, "Mobile Robot Navigation using the Bag-of-Words Algorithm" `http://www.hilandtom.com/thesis.pdf`

Abstract

This paper describes BoWSLAM, a scheme for a robot to reliably navigate and map *a priori* unknown environments, in real-time, using monocular vision alone. BoWSLAM can navigate challenging dynamic and self-similar environments, and can recover from gross errors. Key innovations allowing this include new uses for the Bag-of-Words image representation; this is used to select the best set of frames to reconstruct positions from; and to give efficient wide-baseline correspondences between many pairs of frames, providing multiple position hypotheses. A graph-based representation of these position hypotheses enables the modeling and optimisation of errors in scale in a dual graph, and the selection of only reliable position estimates in the presence of gross outliers. BoWSLAM is demonstrated mapping a 25 minute 2.5km trajectory through a challenging and dynamic outdoor environment without any other sensor input; considerably further than previous single camera SLAM schemes.

1 Introduction

This paper describes BoWSLAM: a scheme allowing a mobile robot equipped with a single camera and moving in three dimensions to position itself in a previously unknown environment. BoWSLAM uses new techniques to successfully navigate dynamic environments despite erratic camera motion, corrupted position estimates, total disorientation, or large accumulated errors. A map of the environment that has been explored is built, and this map is refined and improved when places are re-visited.

The most successful schemes to-date for positioning using a single camera (Clemente et al., 2007; Lemaire and Lacroix, 2007; Eade and Drummond, 2008; Nistér et al., 2006; Dellaert and Kaess, 2006; Mouragnon et al., 2009; Strasdat et al., 2010) have been demonstrated navigating tracks of a few hundred metres at most, through mostly static environments. BoWSLAM is demonstrated mapping a 2.5km trajectory in real-time through a dynamic environment, with no other sensor or odometry input. BoWSLAM is designed to enable the positioning of low-cost robots with poor odometry or unpredictable motion, for example walking robots or quadrotors, or to aid pedestrian navigation.

Two main innovations allow BoWSLAM to navigate difficult environments: firstly the Bag-of-Words (BoW) algorithm (Sivic and Zisserman, 2003), normally used for detecting 'loop closure' by recognising when locations are re-visited, is used in two new ways: for fast feature matching, allowing multiple position hypotheses to be generated for each camera location, and for smart selection of frames from which to compute each pose. Secondly a graph-based map of these relative position hypotheses allows the selection and optimisation of only the most reliable position estimates, and allows the perceived scale of the world to be modelled and scale-drift to be corrected via a dual graph.

BoWSLAM is substantially different from any previous single camera navigation scheme: a high framerate is not required, enabling a high-level representation of each frame to be built and multiple position hypotheses to be calculated; few assumptions about motion are made, allowing navigation despite erratic camera motion or gross errors in dynamic environments, and a novel graphical representation of relative positions and scales allows selection of only the best position hypotheses in the presence of many gross outliers.

This paper is organised as follows: the next section describes contemporary approaches to navigation and SLAM using vision and the BoW algorithm; Section 3 describes BoWSLAM, the new uses for the BoW algorithm, and how information from the additional position hypotheses it generates is used; Section 4 presents results from positioning a camera with videos from difficult indoor and outdoor environments; and the final section discusses our conclusions and future plans. Full source code, and videos of BoWSLAM in operation, are available online at `http://hilandtom.com/tombotterill`.

2 Background

Low-cost positioning in dynamic real-world environments is an essential enabling technology for mobile robots working in homes, workplaces, and outdoor environments where GPS is unreliable or unavailable. A single camera is an ideal sensor for this as it is inexpensive, passive, compact, non platform-specific and requires relatively little power, however existing schemes for navigating using monocular vision are prone to failure due to gross errors or disorientation in difficult environments. In addition position estimates obtained by accumulating many small relative position measurements (dead-reckoning) from sensors such as vision will accumulate errors and drift over time, eventually rendering them useless. For long-term positioning the environment being explored must also be mapped so that the robot can recognise previously visited locations and correct errors in its position estimate (Smith et al., 1990). This combination of positioning and mapping is known as Simultaneous Localisation and Mapping, or SLAM.

Many existing real-time systems for visual navigation suffer from larger accumulated errors than systems using other sensors or sensor combinations (laser scanners, inertial measurement units, wheel encoders), and often encounter gross errors or fail entirely in visually difficult environments (for example dynamic environments). These problems are caused by data association difficulties (Eade and Drummond, 2008; Bosse et al., 2004), large non-Gaussian uncertainties in estimated point depths (Montiel et al., 2006), moving features incorrectly identified as being static, and poorly conditioned geometry. An additional problem with monocular vision is a global scale ambiguity: the actual scale of the world and speed of the camera can only be estimated by identifying something of known size, and small errors in scale can accumulate over time. However vision has a key advantage over other sensors: rich information about the environment is collected that may be used to recognise when places are re-visited, even when the robot is lost. This ability to detect loop closure allows accumulated errors to be corrected, enabling accurate long-term positioning in bounded environments.

Recently several visual navigation systems have been developed addressing some of these problems (Nistér et al., 2006; Clemente et al., 2007; Eade and Drummond, 2006, 2008; Mouragnon et al., 2009; Strasdat et al., 2010). These all work by identifying a stationary 3D world and estimating both the position of landmarks and the camera within it. These systems fall into two categories, firstly Visual

SLAM schemes that estimate a global map while positioning the camera in this map, and secondly Visual Odometry schemes performing dead-reckoning by vision. The next two sections detail these different approaches.

2.1 Visual Simultaneous Localisation and Mapping

SLAM involves the simultaneous estimation of a map and the robot pose; the two most popular contemporary approaches to this estimation are to use either an Extended Kalman Filter (EKF; Dissanayake et al., 2000) or a particle filter (Montemerlo et al., 2002). These approaches can be very successful on a small scale, but have a limited ability to navigate larger environments or assimilate loop closure information. Both approaches have been adapted for SLAM using a single camera; here we describe some of the most successful of these schemes.

MonoSLAM (Davison, 2003) is an EKF-based SLAM scheme using monocular vision. Landmark positions (image feature locations) are estimated directly from observations in the image using an EKF. Several extensions have been proposed including the Inverse Depth representation to reduce errors from initially-uncertain landmark depths (Montiel et al., 2006) and active landmark identification (loop-closure detection) to recover from disorientation (Williams et al., 2008), but despite these improvements real-time performance is limited to maps of a few tens or hundreds of landmarks (Williams et al., 2008). Where there is no loop closure, the small number of landmarks tracked in each frame, and linearisation errors from the EKF, lead to significant drift. Erratic camera motion may be reconstructed with very high frame rates (200Hz) (Gemeiner et al., 2008), however this limits 'real-time' performance to just a few seconds.

Clemente et al. (2007) extend MonoSLAM to larger environments by using submaps: MonoSLAM is used on a small scale, then a new submap is started when this map grows large. The scheme by Estrada et al. (2005) is used to combine these locally-accurate submaps into an accurate global map, by optimising transformations between submaps. A 250m loop is closed, giving an accurate global map. Validating the mutual consistency of tracked features before incorporating them into the EKF ensures that positioning does not fail, despite a dynamic environment including pedestrians.

Another EKF-based approach by Lemaire and Lacroix (2007) combines bearings to landmarks with a prior depth distribution in an EKF, which refines these landmark position estimates as more observations are made. Panoramic images give a wide range of bearing measurements to each landmark, and provide the same field of view on repeated visits to a location enabling loop closure to be detected reliably. Navigation of a 200m trajectory is demonstrated, however this is unlikely to scale much further due to the EKF's poor complexity and potential failure if any outlier measurements are incorporated. Panoramic cameras combined with robot odometry are also used by Dellaert and Kaess (2006), with measurements integrated in a 'square root filter'; this has lower complexity than an EKF (the linear system to be solved is kept sparse); a 200m indoor trajectory is mapped in 11 minutes, although no loops are closed.

An alternative estimation approach is given by the FastSLAM particle-filter (Montemerlo et al., 2002); this has been adapted for monocular vision (Eade and Drummond, 2006), but real-time operation is limited to a few hundred landmarks, and loop closure is demonstrated only on small scales.

Several recent SLAM schemes avoid some of the difficulties in scaling to large environments with graph-based representations of the world: as the SLAM problem consists of constraints (measurements) between robot poses and observations of the world then if these constraints are represented as the edges in a graph then a solution is given by an optimisation on this graph (Thrun and Montemerlo, 2006). A popular and scalable graph-based formulation is given by the Atlas framework (Bosse et al., 2004). In this hybrid approach each node represents a small group of landmarks in their own local coordinate frame (a 'submap' where a conventional SLAM algorithm is used), and each edge represents a transformation between these coordinate frames. Accurate positions of a node relative to a root node can be reconstructed cheaply by finding the 'shortest' path through the graph, where edges are weighted by a measure of their transformation's accuracy.

The most efficient algorithms to perform the global optimisation needed to generate world maps from these graph-based representations first assume the errors in these transformations are Gaussian and independent, then iteratively update positions to reduce local errors, either via a 'relaxation' procedure (Frese et al., 2005), or by stochastic gradient descent, with updates applied to consecutive subsections of the robot's trajectory (Olson et al., 2006). The TORO framework (Grisetti et al., 2007b) efficiently implements a stochastic gradient descent optimisation where robot poses are grouped spatially rather

than temporally. A 3D generalisation of TORO (Grisetti et al., 2007a) works by alternately optimising rotations and translations. This approach allows good approximations to the maximum likelihood map to be found efficiently even in large maps, and if the number of cycles is limited then the number of iterations needed is constant and each iteration has linear complexity in the size of the mapped area (Frese et al., 2005).

An Atlas-based single camera SLAM scheme is described by Eade and Drummond (2007). The transformations and estimated covariances between nodes are refined by gradient descent subject to consistency around cycles. Large loops can be detected using BoW and closed rapidly by adding edges to this graph (Eade and Drummond, 2008), although optimisation becomes expensive when maps are large (Eade, 2008).

Two more graph-based schemes using stereo cameras are also relevant: Konolige et al. (2009) accumulate a sequence of relative poses between frames, together with relative poses from global loop-closure detection, allowing recovery from gross errors. Navigation is demonstrated over 10km, although relative positions from an IMU are needed for about 2% of frames where positioning fails due to camera occlusion. The authors claim their scheme is applicable to single-camera-SLAM, although this is not demonstrated, and no framework to model scale-drift is proposed. Again a TORO-like optimisation framework is used to produce optimal global maps. A similar graph-based map is built by Newman et al. (2009), using stereo vision and a laser scanner. Every pose is represented by a node, and edges represent relative positions of nodes. Real-time performance is possible until a global map is required; this global optimising is a slow off-line process, taking 20 minutes for a 50 000 pose graph.

An alternative to these global mapping approaches is described by Mei et al. (2009): SLAM with a stereo camera is demonstrated on a trajectory of over a kilometre but only local submaps consisting of nearby nodes are optimised (using a local bundle adjustment), keeping complexity to constant time. A similar scheme for single camera SLAM is described by Strasdat et al. (2010). Relative poses between a subset of frames are found by a local bundle adjustment, then the pose graph of these relative poses is refined by Levenberg-Marquardt nonlinear optimisation. Notably, relative scale estimates between frames are also optimised. When the camera travels around a 220m loop, significant scale drift is observed (a three-fold difference when the loop is closed); this is corrected when the loop closure is detected, greatly reducing distortions in the map.

2.2 Visual Odometry

Visual Odometry is a slightly simpler problem than SLAM in that only the camera motion is of interest, and no global map is built. This frees computational resources for the computation of accurate pose estimates, however without the loop closure detection of SLAM small errors in position will accumulate over time and grow without bound, until eventually the true pose is unknown. One successful visual odometry scheme is that by Nistér et al. (2006), which uses structure-from-motion techniques to position a monocular camera in static outdoor environments over a 600m trajectory. Approximate relative poses over three frames are estimated and refined iteratively, then 3D world points are estimated. Several additional frames can then be positioned with respect to this local structure. The scheme is dependent on features moving typically less than 10% of the frame width between consecutive frames, and on features being visible for many consecutive frames. Accumulated errors are as low as 2% of the distance travelled after two thousand frames, although without a map there is no means to correct these errors, or to recover from gross errors. More recently Mouragnon et al. (2009) use incremental bundle-adjustment to reconstruct camera trajectories of up to 500m in dynamic environments, after outlier removal using RANSAC, although again feature motion is constrained.

2.3 The Bag-of-Words algorithm in Visual Navigation

Eventually a robot travelling in an unknown environment will re-visit somewhere it has been before. Detecting this loop-closure event is essential for long term navigation, and to recover from gross errors or disorientation. In addition knowledge of the loop allows map estimates around the loop to be improved and a consistent map to be maintained. Vision is an ideal sensor for detecting loop closure as it captures rich and distinctive information about the environment, and the most successful schemes for loop closure detection are based on the Bag-of-Words (BoW) algorithm. The BoW image representation describes an image by the set of features that it contains. A set of feature descriptors is extracted from each

image and each descriptor is mapped to its closest match from a quantised 'dictionary'—a discrete set of representative features (also known as a 'codebook' or 'vocabulary'). The image BoW algorithm was developed for fast and accurate image search—images can be compared very quickly to see if they have many features in common, and hence may be of the same object or location. This is often highly accurate despite ignoring the image geometry. The BoW algorithm's ability to recognise the location a photo was taken has been demonstrated by Sivic and Zisserman (2003), who used it to segment movies by location, and by Cummins and Newman (2008a, 2009) who accurately detect loop closure events in a database of 100 000 frames collected over hundreds of kilometres.

Most BoW schemes generate a dictionary off-line by K-means clustering of descriptors from a training set (e.g. Nistér and Stewénius, 2006; Newman et al., 2006; Sivic and Zisserman, 2003; Li and Perona, 2005). An alternative approach is to dynamically construct a dictionary from the features which are encountered as the environment is explored, so that environments with features not represented in the training set can be recognised effectively. Such a scheme is described by Angeli et al. (2008a), which closes small loops indoors, although real-time (1Hz) performance is limited to about 2000 images. Similarly Eade and Drummond (2008) and Nicosevici and Garca (2009) initialise new words dynamically and add them to a vocabulary, although in both cases the number of words used are orders of magnitude smaller than other schemes (a few thousand).

SIFT (Lowe, 1999) and SURF (Bay et al., 2008) are the most popular descriptors for BoW scene recognition (Nistér and Stewénius, 2006; Newman et al., 2006; Cummins and Newman, 2008b; Angeli et al., 2008b,a; Csurka et al., 2004; Li and Perona, 2005; Filliat, 2007; Eade and Drummond, 2008; Nicosevici and Garca, 2009). When used for robot localisation these schemes typically index images at around 1Hz (Cummins and Newman, 2008b; Angeli et al., 2008b), as extracting SIFT or SURF features from every frame is too slow for real-time processing (Rosten and Drummond, 2006; Cummins and Newman, 2009). Therefore when used for loop-closure detection in SLAM, BoW is usually run in isolation on a subset of frames (Eade and Drummond, 2008; Newman et al., 2006, 2009). Occasionally a good loop closure candidate is found and this information is used to update the map. Some of these schemes explicitly estimate the loop closure probability (Angeli et al., 2008a; Cummins and Newman, 2008a, 2009), whereas others achieve high levels of accuracy with heuristic measures of relative image similarity (Newman et al., 2006; Nistér and Stewénius, 2006).

2.4 Summary of prior research

In summary, numerous schemes have been demonstrated which enable robots to position themselves using a single camera alone; these schemes often work in real-time and over tracks of hundreds of metres, while often producing accurate maps within a few percent of ground truth. There are two major limitations of these approaches however: firstly the environments explored are largely static, and rapid cornering (during which single-camera geometry is poorly conditioned) is largely avoided, however typical indoor or urban environments where robots will one day operate contain moving people and vehicles; a robot moving through these environments must cope not only with these dynamic objects, but also with erratic motion including rapid cornering. To overcome these limitations a successful SLAM scheme must function despite these challenges, and critically, must be able to recover from gross errors in positioning.

The second limitation of many single camera SLAM schemes (Lemaire and Lacroix, 2007; Dellaert and Kaess, 2006; Eade, 2008; Strasdat et al., 2010) is their high algorithmic complexity—the SLAM algorithms soon become too costly for real-time operation as the mapped area grows. SLAM methods with lower cost and complexity should be used for longer term navigation, for example the pose-graph or submap methods of Mei et al. (2009); Estrada et al. (2005).

The second important observation is that the BoW algorithm is generally very successful at efficiently recognising when two images show the same scene. This observation is key to the operation of BoWSLAM, our new SLAM scheme which is designed to overcome these limitation and is described in the following section.

3 The BoWSLAM Single Camera SLAM Scheme

The previous section described contemporary approaches to single-camera SLAM, and their limitations. This section describes our new scheme, BoWSLAM, which uses new techniques enabled by adding every

image to a BoW database to overcome some of these limitations and to allow robust positioning in visually challenging dynamic environments. In this section we first give an overview of BoWSLAM, then describe its individual components in detail.

All previous visual navigation systems operate in basically the same way: the camera captures an image showing part of the same environment that was viewed previously, matches are found between features in the new image and the same features in previous images, the camera's new pose is calculated from these matches, based on the assumption that some of the environment is static, and finally a map is built from the observed features. BoWSLAM's basic mode of operation is exactly the same; in addition to allow long-term robot positioning in real-world environments, where moving objects and erratic camera motion make navigation difficult, we have the following requirements:

Active loop-closure detection To recover from gross errors, and to maintain a consistent and accurate map for extended periods, the robot must detect when known locations are re-visited, without any prior knowledge of its position.

Wide-baseline matching The capability to register frames captured from significantly different viewpoints is essential where sequences of frames are unusable due to occlusion, motion blur, or erratic camera motion, and following loop-closure.

Robust estimation Position estimation must be robust in the presence of moving objects and in self-similar environments.

Multiple position hypotheses Despite robust position estimation, errors will inevitably still occur. The ability to reject these erroneous position estimates is essential for long-term navigation.

Active loop-closure detection refers to searching the entire map for locations which could be the current location, so that relocalisation is possible even when the robot is lost, or its position estimate contains large errors. The leading framework for active loop-closure detection is the BoW algorithm, and to use this some frames must be represented as 'Bags-of-Words'. When two frames are matched by the BoW algorithm, wide-baseline correspondences between them can be obtained cheaply and effectively from their BoW representations. As wide-baseline matching allows us to efficiently register any pair of overlapping frames, we can use any such sequence of frames to position the camera. Frames from which to compute the relative position of the latest frame are selected efficiently using the BoW algorithm and include loop-closure candidate frames from previous visits to a location, and the most suitable recent frames, hence allowing recovery after periods of erratic motion or occluded frames. Our BoW scheme, and these new applications, is described in Section 3.1.

When a frame is captured, feature points are detected and descriptors of the image around these points are extracted (we use simple image patches centred on corners found using the FAST corner detector; Section 3.1). These descriptors are used to describe the frame as a BoW. We now use the BoW database to find both nearby frames, and feature matches with these frames. Given a correct set of matches between frames, we can compute the relative pose of the cameras via a least-squares approach. However these feature matches are often contaminated with large numbers of outliers, especially when moving objects are present or overlap between frames is small. These outliers must be removed before a least-squares approaches can be applied. A popular method for removing these outliers is the RANSAC framework, in which the camera pose is computed while simultaneously identifying an inlier set. We use the BaySAC framework, a RANSAC variant allowing large numbers of low-quality correspondences to be used efficiently, followed by linear and nonlinear least-squares refinement, to compute the camera pose. These procedures for computing and refining the relative camera pose are described in detail in Section 3.2.

Given these relative pose estimates between pairs of frames we can reconstruct the camera's trajectory by adding a sequences of these relative poses together. Any errors in one pose however will also affect subsequent pose estimates. While small accumulated errors can be dealt with via the SLAM framework (Section 3.4), gross errors will inevitably also occur. In order to maintain consistent maps and accurate pose estimates it is essential to avoid using these erroneous pose estimates.

To eliminate erroneous pose estimates we compute the pose of each frame relative to each of several nearby frames, leading to a graph of pose estimates (Figure 8). Two frames can be positioned relative to each other in many different ways via all of the different paths between them. From this graph we choose a small subset of possible position hypotheses containing only the most reliable position estimates,

hence minimising the chance of incorporating erroneous position estimates into our map. This graph of multiple position hypotheses, and the dual graph in which we model the scale drift that occurs in single camera SLAM, are described in detail in Section 3.3.

Finally, in order to create a globally accurate map, we optimise a graph of independent relative position estimates using the efficient TORO framework. The TORO framework, and the generation of TORO's input graph are described in Section 3.4.

In summary, BoWSLAM works by representing every frame as a BoW, then using the BoW representation to select multiple nearby frames from which to compute relative positions. The latest frame is positioned relative to each of these frames using the robust BaySAC framework together with correspondences from the BoW algorithm. A subset of these multiple position hypotheses that is unlikely to contain any gross errors can then be selected and refined to accurately position the camera in a global map. An overview of the BoWSLAM algorithm is given in Figure 1, and the following sections describe each component in detail.

For each frame:

1. **Add new frame to BoW database:**
   - Capture new frame from camera
   - Detect corners and extract features
   - Add image to BoW database
2. **Select recent frames for relative positioning, and potential loop closure candidates. For each of these frames:**
   - Find BoW feature correspondences with current frame
   - Compute pose relative to several previous frames
   - Reconstruct 3-D landmark positions (up to scale ambiguity)
   - Align landmark positions with earlier observations to resolve scale change and reconstruct camera positions
3. **Update graph to select best pose estimate for every frame**

Figure 1: Overview of the BoWSLAM Algorithm

3.1 Feature description and the Bag-of-Words image representation

To reconstruct the position of a frame relative to previous frames we must establish matches between features in this frame and the same feature seen in the previous frame. The features we match are corner features found using the FAST corner detector (Rosten and Drummond, 2006); these provide good repeatability and localisation accuracy while being cheap to extract, taking just 10ms for a 640×480 image. The highest-scoring FAST corners are chosen subject to a minimum separation constraint (in experiments the minimum separation constraint degrades repeatability slightly but is needed to ensure geometry is well conditioned). To describe the image around these corners we use simple image patches. Typically patches sized 33×33 pixels are down-sampled to 11×11. The Euclidean (L_2) distance between a pair of these descriptors provides a good measure of their similarity. Examples of the features extracted are shown in Figure 2.

BoWSLAM uses the BoW scheme described by Botterill et al. (2008). Like other BoW schemes each image is represented by the set of descriptors that have been extracted from it, each quantised to the most similar-looking of a representative set of descriptors (Figure 3). This set of representative descriptors (a 'dictionary' of 'image words') is chosen so that the same feature viewed in multiple images should map to the same image word each time, and different-looking features should map to different image words. Two images with many image words in common contain many similar-looking features, and therefore are likely to show the same place. These matches can be found very rapidly as comparing

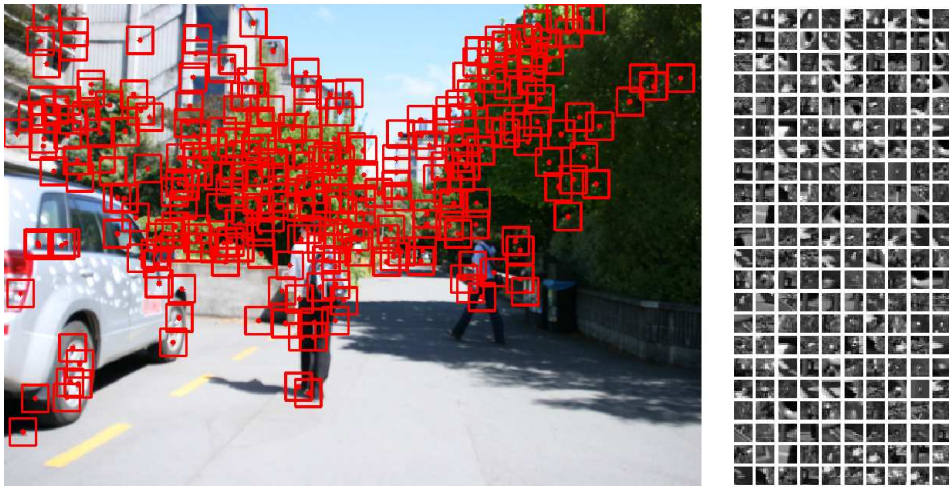


Figure 2: Interest points found by the FAST corner detector. A minimum separation constraint is needed to ensure geometry is well conditioned. Image patches centred on each corner are used as descriptors.

images consists simply of comparing their BoW representations, which are essentially sparse vectors of each word's frequencies'.

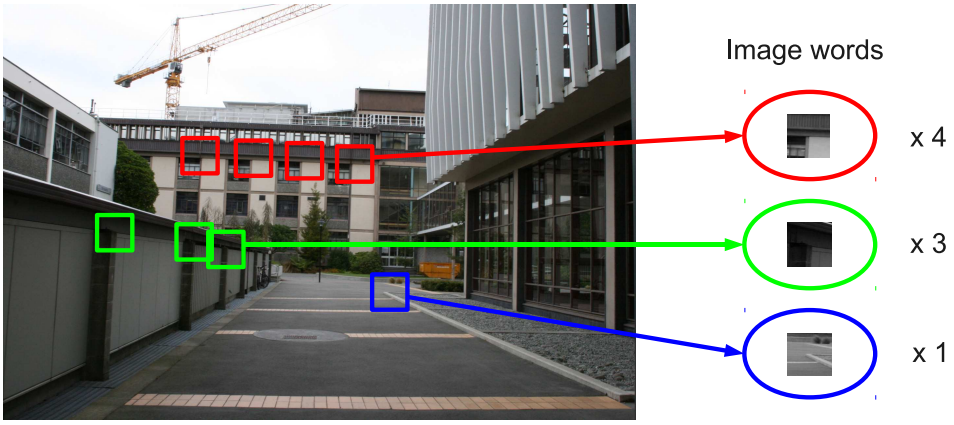


Figure 3: To represent an image as a BoW, we map each descriptor to the nearest image word (the cluster centres found previously). For each image typically 300 features are mapped to about 150 image words, out of a total of tens or hundreds of thousands.

A suitable dictionary of image words can be found by clustering descriptors extracted from a training set of images, so that each cluster corresponds to a particular class of features which should be described by the same image word. The cluster centres are then used as image words. To cluster the training set into image words an algorithm such as Lloyd's algorithm for k-means clustering is usually used, however this training is slow (taking hours for a reasonably-sized training set) and will work well only in an environment resembling the training data. Once the dictionary is built there is also the problem of efficiently finding the closest image word to a particular descriptor.

To solve some of these problems Nistér and Stewénius (2006) propose a hierarchical dictionary: the training set is clustered into a small number of image words, then each cluster is clustered again into a small number of smaller clusters (Figure 4). This is repeated until a sufficient number of image words is found (for example 10^5 image words if we clustered descriptors into ten subsets at each of five levels). Clustering is more efficient overall as we are clustering into a smaller number of subsets each time, and assigning a descriptor to the closest image word is fast: the closest centre at the top level is found, then

the closest centre from its child clusters is found, etc. until the closest image word is found at the bottom level.

A further advantage of this hierarchical dictionary is that when a large number of descriptors are assigned to a small number of centres, clustering a small subset of the descriptors usually gives a good clustering for the entire set. This observation reduces the time needed to generate an effective dictionary, allowing new dictionaries to be generated dynamically by clustering the descriptors that have actually being observed, and hence which will effectively describe the environment that has being explored (Botterill et al., 2008).

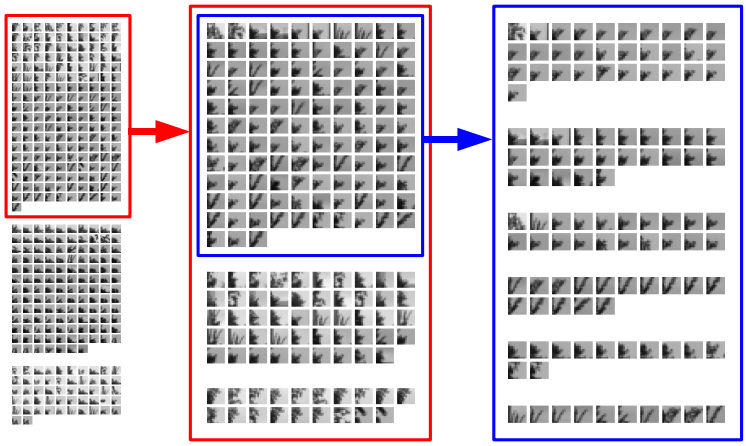


Figure 4: Hierarchical clustering: descriptors (image patches in this case) are clustered into a small number of large clusters using a fast approximate clustering algorithm. Each cluster is then clustered again into smaller clusters. Typically this is repeated at about six levels; an example of clusters at the bottom three levels is shown here—the descriptors on the left are clustered into three smaller clusters, each of these smaller clusters is clustered again (the first of these is clustered again into the three smaller clusters shown in the centre), then each cluster is clustered again; the first one gives the six clusters shown on the right. Each of these cluster's centres' forms a image word.

Some image words are more useful than others for identifying if two images show the same place, for example most images in the dataset in Figure 10 contain image words describing trees and buildings, but only a few contain a 'Give Way' road sign, therefore two images containing words describing the Give Way sign are likely to show the same place. To take this distinctiveness into account we weight the images' word-frequency vectors with the heuristic Term Frequency-Inverse Document Frequency (TF-IDF) score defined as:

$$w_i = \log \frac{T}{C_i} \tag{1}$$

where w_i is the weight of image word i calculated from T, the total number of images, and C_i the total number of images where image word i occurs (Nistér and Stewénius, 2006).

The most similar images to a query image are now found by iterating through the BoW database and comparing each images' weighted word-frequency vector with that of the query image. The images with the most similar weighted word-frequency vectors are most likely to show the same place. Recognition accuracy is high despite geometric information being ignored, and results are robust to small changes in the scene, due to the large number of features that are compared.

3.1.1 Correspondences from the Bag-of-Words algorithm

A corner feature visible in two frames normally results in each frames' BoW representation containing the same corresponding image word. If there is exactly one of a particular word in each image then this probably corresponds to a matching feature (as used by Cummins and Newman, 2009, for validating detected BoW matches). If there are a small number (N and M) of a particular word in two images, all

possible pairs can be considered as candidate matches ('$N - M$ correspondences'). BaySAC (Botterill et al., 2009b) can then be used to choose the subset of these that are correct (described in Section 3.2.1).

This BoW feature matching is very fast, however variability in cluster sizes leads to poorly conditioned correspondences being introduced, while good correspondences are missed when descriptors are split between two nearby clusters. Typically only 40% of the correspondences found by a brute-force matching are found by BoW feature matching. To avoid this problem we use the coarser partitioning of the descriptor space at a higher level of the hierarchical dictionary, and find matches within each cluster by brute-force descriptor comparison. This is not prohibitively expensive and typically over 80% of the correspondences found by a brute-force approach are found, with very few additional outliers. $N - M$ correspondences are used where a feature is sufficiently similar to several others.

Indexing an image with 300 features takes typically 2 milliseconds, and finding correspondences 2 milliseconds, whereas a brute-force matching approach takes 50 milliseconds. BoW feature matching gives the biggest gains when each image is compared to many others, many landmarks are observed, and high-dimensional descriptors are used; all are the case in BoWSLAM.

This BoW feature matching method is similar to the kd-tree approach of Beis and Lowe (1999) which is often used for matching SIFT descriptors (e.g. Cummins and Newman, 2009). BoWSLAM uses the partitioning found by clustering descriptors rather than an arbitrary partitioning about descriptor components. Clustering descriptors minimises the number that will fall near the boundary of two clusters, so unlike the kd-tree approach, by choosing a sufficiently coarse partitioning we do not need to search neighbouring bins for matches.

Other visual navigation schemes (Nistér et al., 2006; Davison, 2003; Mouragnon et al., 2009) reject candidate correspondences with multiple potential matches, and/or limit the image area that is searched for each; however this is not necessary in BoWSLAM where $N - M$ correspondences are used. This allows BoWSLAM to register each frame to any other frame with many features in common, even in self-similar environments or when the viewpoint is significantly different.

3.2 Computing relative positions

This section describes how BoWSLAM computes the position of a camera based on a sequence of frames (the selection of this sequence is described in Section 3.3). First the relative position and orientation of two frames is estimated (Section 3.2.1). Next, the 3D positions of landmarks observed in these frames are reconstructed. Finally the true scale of these landmarks, and hence the true motion of the camera between frames, is computed by aligning these landmark positions (Section 3.2.2).

3.2.1 Relative camera pose computation

The relative pose of two cameras consists of a 3D rotation and translation. From correspondences between matching features in two frames we can compute this rotation and the direction of the translation, however the magnitude of this translation cannot be determined without further information.

Algorithms to compute the relative pose from point-correspondences usually work by estimating the essential matrix E (Hartley and Zisserman, 2003, chapter 9), a 3×3 matrix encoding the rotation and translation direction with the property that, for a noiseless correspondence between points $\mathbf{p}$ and $\mathbf{p}'$ in the image ($(p_x, p_y, 1)$ and $(p'_x, p'_y, 1)$ as homogeneous coordinates):

$$\mathbf{p}'^T E \mathbf{p} = 0 \tag{2}$$

At least five correspondences are needed to estimate E; given exactly five correspondences there are up to ten possibilities for E, which can be computed from Equation 2 using the method by Stewénius et al. (2006). E is uniquely determined by eight correspondences, and can be computed from the matrix F that minimises

$$\sum_{i=1}^{C} \mathbf{p}_i'^T F \mathbf{p}_i \tag{3}$$

for $C \geq 8$, using the normalised eight-point algorithm described in Hartley and Zisserman (2003); chapter 11. This linear least-squares method minimises the effects of small errors in point localisation, however it is sensitive to incorrect correspondences (outliers), which can lead to a completely incorrect solution.

Outliers are common in BoW correspondences due to mismatched features, self-similar environments, and moving objects; for this reason inliers must first be identified.

To compute E while simultaneously identifying inlier correspondences BoWSLAM uses the BaySAC framework (Botterill et al., 2009b). BaySAC is a variant of the popular RANSAC framework for fitting a model to data points contaminated with many gross outliers (Fischler and Bolles, 1981). In RANSAC many small hypothesis sets of data points are chosen randomly. Each hypothesis set is used to generate models. Each model is compared to every data point, and the model consistent with the largest number of data points is selected. Assuming this model is correct, these data points are inliers. RANSAC can be slow to find the correct model however when inlier rates are low, and many data points are poor-quality. BaySAC attempts to solve this problem: instead of choosing hypothesis sets at random, BaySAC aims to choose the hypothesis set most likely to consist only of inliers at each time. This hypothesis set is chosen by estimating the prior inlier probability for each data point, then at each iteration choosing the data points with the highest prior probabilities (assuming data points' inlier probabilities are independent). Inlier probabilities are then updated as hypothesis sets are tried, and are found to be contaminated by outliers.

To estimate E using BaySAC we first estimate each correspondences' inlier probability by assuming each point has an equal probability p of having a match in the other image, then dividing this probability amongst all $N - M$ matches which include this point (giving each correspondence probability $p/\max(N, M)$). BaySAC then chooses hypothesis sets of five correspondences, calculates all essential matrices compatible with each hypothesis set and considers each as a candidate for the correct model. When choosing hypothesis sets and counting inliers the constraint that each feature has at most one correct match in the other image is imposed.

The essential matrix and inlier correspondences are further refined using top-down outlier-removal (Rousseeuw and Leroy, 1987): the essential matrix fitting all inlier correspondences is computed using the normalised eight-point algorithm, then the correspondences least compatible with this matrix are removed. The essential matrix is decomposed to give four possibilities for the relative camera poses; the correct relative pose is chosen to be the one leading to most reconstructed points falling in front of the cameras.

Equation 3 is a poor measure of the error in relative pose however, as it is biased by the distances of points from the image centre. Ideally we should find the transformation and 3D structure that minimises the distances between where 3D points are projected into the image and where they are measured to lie (the reprojection error). This is costly to compute however, instead we minimise Sampson's approximation to the reprojection error (Hartley and Zisserman, 2003, chapter 11):

$$\sum_{i=1}^{C} \frac{\mathbf{p}_i'^T E \mathbf{p}_i}{(E\mathbf{p}_i)_1^2 + (E\mathbf{p}_i)_2^2 + (E\mathbf{p}_i')_1^2 + (E\mathbf{p}_i')_2^2} \tag{4}$$

The relative pose is refined to give the essential matrix minimising Equation 4 by Levenberg-Marquardt non-linear optimisation. In simulated data this reduces errors by about 30%. 3D point positions can now be reconstructed up to an unknown scale factor (Hartley and Zisserman, 2003, chapter 10).

An alternative approach would be to estimate the trifocal tensor connecting each set of three frames (Nistér et al., 2006), or to optimise the transformation over a sequence of frames (Mouragnon et al., 2009). This would result in a small increase in accuracy at the expense of considerably more computation in BoWSLAM, as many different positions hypotheses are computed from a large number of possible triples of frames (Section 3.3).

3.2.2 Resolving Scale Change

Given a feature viewed in two frames, and the relative pose between the two cameras, the 3D position of that feature can be reconstructed in the local coordinate frame of one of the cameras. As the cameras' relative position is known only up to an unknown scale factor, this feature's 3D reconstruction is also known only up to an unknown scale factor. If the feature is also viewed in an earlier frame we can align this reconstructed feature with its earlier reconstructed position, hence calculating the ratio of these scale factors (Figure 5). This scale factor is proportional to the distance between the two camera positions (the 'baseline length'), so estimating it relative to the previous scale factor gives us the distance moved by the camera relative to the camera's previous motion.

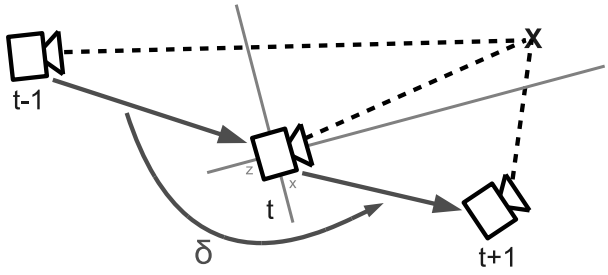


Figure 5: The point X is observed at time t and $t+1$. We know the relative pose of these cameras,up to an unknown scale factor, so can reconstruct X in the coordinate frame of camera t, up-to an unknown scale factor. If this point was also observed in an earlier frame $(t-1)$, we can calculate the ratio of the two scale factors, δ, by aligning the points' 3D reconstructions.

In the noise-free case, two reconstructions of the same feature in the same coordinate frame, X_i and X'_i, will fall on the same ray, i.e. $X_i = \delta X'_i \ \forall \ i$, where δ is the ratio of the two baselines, however in practice reconstructed points contain substantial errors from several sources, including small errors in localising features in the image, amplified by stereo reconstruction; small errors in computing the cameras' relative poses; and from incorrectly matched features that have not been discarded, as they were approximately compatible with the relative poses computed earlier. To estimate the scale change accurately despite these errors we must combine multiple measurements while discarding any outliers.

Given multiple matches between reconstructed 3D points, we first remove outliers by discarding point pairs not within a certain angle of each other, as these do not fall on the same ray. We then calculate the scale factor for each pair X_i, X'_i, giving a set of K scale measurements $\{\delta_i, i = 1, ..., K\}$.

The change in scale between two frames is modelled with a lognormal distribution, $\delta \sim \text{Log-}\mathcal{N}(d, g^2)$ (defined by $\log\delta \sim N(d, g^2)$ NIST/SEMATECH e-Handbook of Statistical Methods, 2010, section 8.1.6.4). This distribution is a good approximation for observed data (much better than the normal distribution; Figure 6), and has the useful properties that the product and inverse of lognormally-distributed variables are also lognormally distributed. d and g are estimated from $\{\log\delta_i\}$ by first removing any remaining gross outliers using Grubb's test (NIST/SEMATECH e-Handbook of Statistical Methods, 2010, section 1.3.5.17), then taking d as the sample mean. As we have assumed $\{\log\delta_i\}$ are normally distributed, then d is approximately normal (d actually has a T-distribution), with variance:

$$g^2 = \frac{1}{K(K-3)} \sum_{i=1}^{K} (\log\delta_i - d)^2 \tag{5}$$

3.2.3 Accumulating relative positions and scales

The previous section describes how to calculate the speed of a camera relative to its previous speed, however its absolute speed is unknown, as a global scale ambiguity exists when navigating using only a single camera (without making assumptions about the speed of the robot or the size of observed objects). For now we assume the first time the camera moved it moved a unit distance; all other motion is now represented in terms of this distance. Other distances can now be calculated by multiplying sequences of relative scales (Figure 7). As these relative scales are modelled as lognormally distributed random variables, then their product s_{jk} is also lognormal:

$$s_{jk} \sim \text{Log-}\mathcal{N}(D_{jk}, G^2_{jk}) \tag{6}$$

$$\log s_{jk} \sim N(D_{jk}, G^2_{jk}) \tag{7}$$

$$\text{where } D_{jk} = D_{ij} + d_{ijk},\ G_{jk} = G_{ij} + g_{ijk} \tag{8}$$

We have now calculated the relative pose of each pair (Section 3.2) and have parametrised the length moved at each time by $s_{jk} \sim \text{Log-}\mathcal{N}(D_{jk}, G^2_{jk})$. Now every frame can be positioned relative to the first frame's position by accumulating these relative pose estimates, with lengths s_{jk} given by the median of the lognormal distribution, $e^{D_{jk}}$. This method allows a map to be built from any sequence of poses.

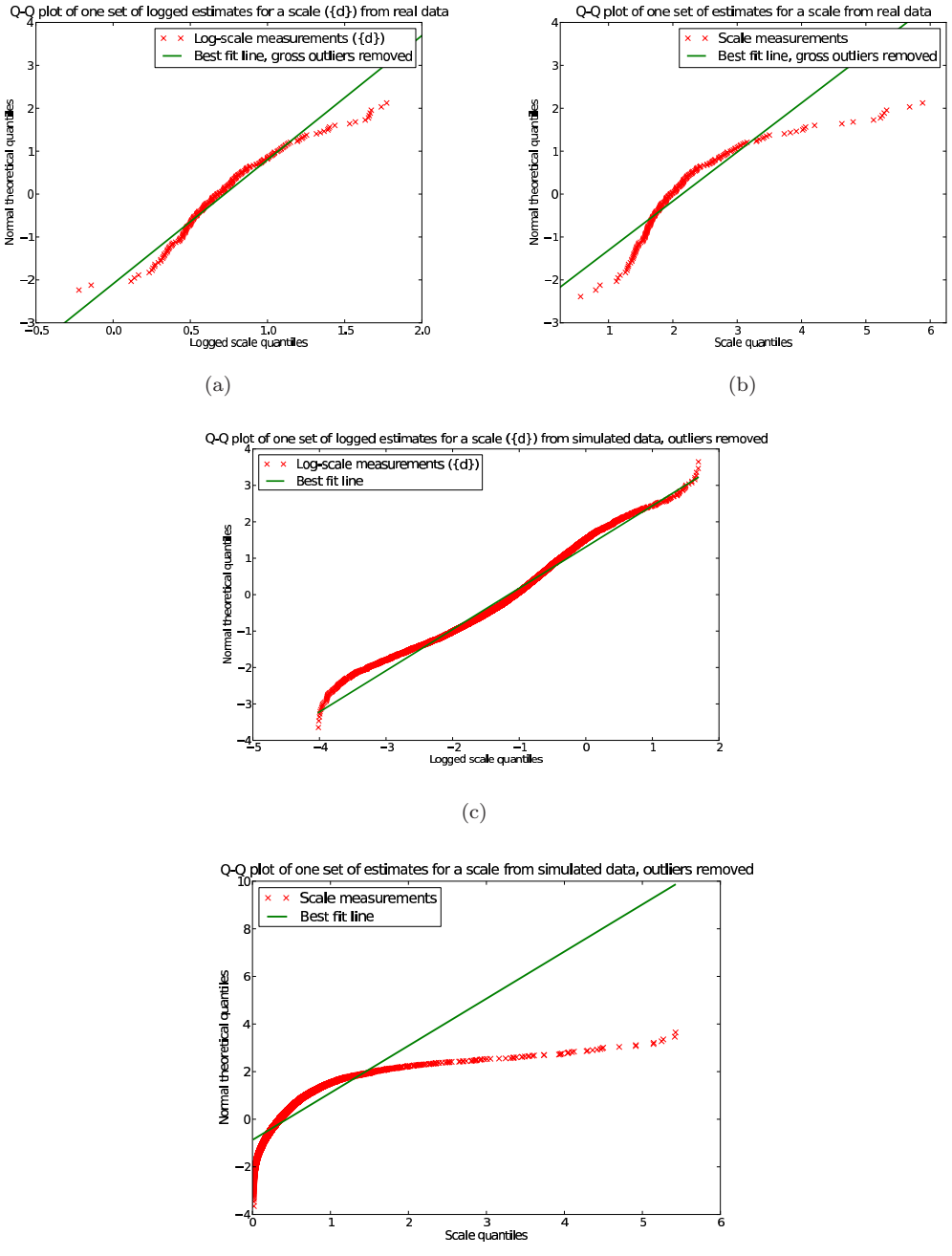


Figure 6: The lognormal distribution is a good approximation for relative scales. Q-Q plots (Thode, 2002) show how well data is approximated by a particular distribution. The quantiles of the data are plotted against the quantiles of a normal distribution; the closer points are to a straight line, the better the approximation. For an example of typical real data the log of the scale measurements (a) is slightly better approximated by a normal distribution than the scale measurements (b); hence the lognormal distribution is a better model. In simulated data, when stereo disparity errors are higher (the mean point depth is 50 times the baseline length in this example) the lognormal distribution (c) is a considerably better approximation than the normal distribution (d). Grubb's test was used to remove gross outliers before making these plots.

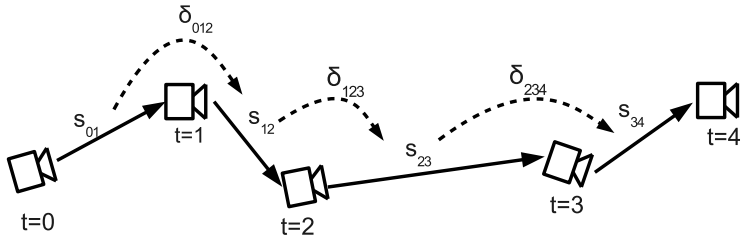


Figure 7: The distance between each pair of cameras j and k, s_{jk}, is calculated by accumulating relative scale estimates, δ_{ijk}. Note that errors in the estimate of δ_{ijk} will accumulate in subsequent scale estimates (and hence scale estimates are correlated). The dashed edges form the edges in the Edge-Vertex dual $E(G)$.

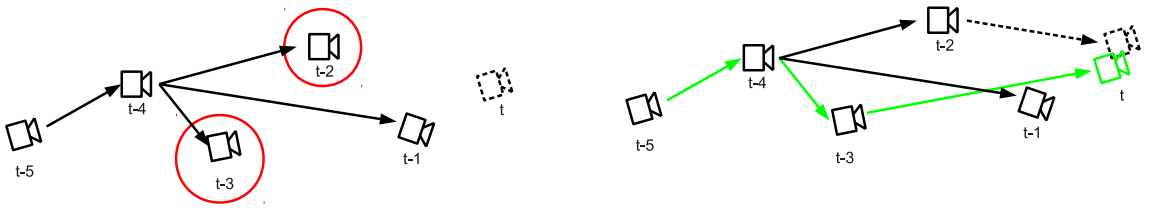
3.3 Graph-based Representation of Multiple Position Hypotheses

This section describes how wide-baseline feature matching and the BoW algorithm allow BoWSLAM to compute multiple position hypotheses for each frame, and how the graph these position hypotheses form is used to select only the most reliable of these position estimates in the presence of gross outliers.

Most contemporary visual navigation schemes track features over consecutive pairs of frames (e.g. Nistér et al., 2006; Davison, 2003; Eade and Drummond, 2008; Strasdat et al., 2010). If camera motion is smooth and the frame rate is high enough there will be large overlap between images, and features move only a short distance, allowing feature tracking by searching a small area where they are predicted to lie. These assumptions are not true in general however; real video sequences often contain sequences of frames with insufficient information for accurate matching (for example blank walls), sequences corrupted by dynamic objects (people walking in front of the camera), or sequences of rapid movement with either motion blur or insufficient overlap for reconstruction. Ideally each frame should be positioned relative to a frame with which it has large overlap (and many of the same landmarks visible), significant camera motion (for well-conditioned reconstruction), and few dynamic objects. These frames are chosen by using the BoW algorithm to select frames with many features in common with the current frame. From these candidate frames we choose those that already have accurate position estimates. This BoW sampling strategy avoids selecting frames to register to that contain too many dynamic objects or are blurred (as these could not be positioned previously), and avoids frames with insufficient overlap as these do not generate a strong BoW match. At the same time potential loop closure candidates are chosen—these must match the current frame at least as strongly as several recent frames. When the camera is stationary, moving slowly, or revisiting an earlier location, the baseline (the distance between two poses) can be too short for accurate 3D reconstruction. In this case no new position estimates are calculated until the camera has moved sufficiently for an accurate position to be calculated. This automatically keeps the map sparse when repeatedly visiting the same place, an important requirement for long-term navigation in a bounded environment.

This strategy results in each camera position being computed relative to several (typically one to four) other camera positions, giving several position hypotheses. These multiple position hypotheses naturally form a graph, with video frames corresponding to nodes and position hypotheses relative to other frames' locations corresponding to edges (Figure 8). Each edge has a set of associated 3D landmarks which are visible in the two frames it connects. Every path connecting two frames in this graph gives a sequence of relative position estimates; the relative position of these two frames is given by accumulating the relative position estimates along one of these paths. This graph is similar to the graph in the Atlas framework (Bosse et al., 2004) with edges representing transformations. In BoWSLAM every video frame corresponds to a node (as in Newman et al., 2009), so that transformations connect robot poses, in contrast to Bosse et al. (2004); Eade and Drummond (2007); Strasdat et al. (2010) where nodes represent local coordinate frames with associated landmarks.

We now have a pose graph, G, with each frame positioned relative to several nearby frames. This provides a large number of possible paths by which two frames can be positioned relative to each other. The Atlas framework provides an efficient way to choose a good path between any two frames by assigning



(a) When a new frame is captured first choose candidate frames from which to compute relative position hypotheses.

(b) Add position hypotheses—the sequence of relative positions most likely to be outlier-free (highlighted) is used to position the new frame; another position hypothesis (dashed) is not used for now.

Figure 8: Camera positions form a graph with edges representing relative position hypotheses. To add a new frame at time t (dashed) suitable candidates to position it relative to are selected (a). The new frame is positioned relative to several other frames leading to several position hypotheses (b). Of these hypotheses the position most likely to be outlier-free is selected. These positions are updated as more position hypotheses become available.

appropriate quality weights to each edge then finding the 'shortest' (highest quality) path between the frames, however in the case of single camera SLAM, estimates of relative scale must also be taken into account. These relative scale estimates are dependent on sequences of relative poses, introducing constraints between pairs of edges in G, which we model using the dual-graph approach described in the following section.

3.3.1 Dual-graph representation of relative scale

When positioning using a single camera, we find the principal sources of error are gross outliers (usually due to dynamic environments or from failing to detect remaining outlier correspondences) and scale drift, as errors in relative positions are proportional to errors in the estimated scale, and errors in orientation are typically low by comparison. When gross errors in relative pose do occur, they normally lead to gross errors in the corresponding relative scale estimate. Therefore we aim to choose the path through the pose graph G that is least likely to contain gross errors in relative scale estimates. Relative scale estimates are not associated with edges in G however; instead a relative scale estimate is associated with every pair of edges meeting at a node (Figure 7), i.e. is associated with edges in a new graph, the 'edge-to-vertex dual' $E(G)$ of the pose graph G (or 'line graph'; Balakrishnan, 1997, chapter 2). Vertices in $E(G)$ correspond to edges in G, and edges in $E(G)$ connect two vertices if the corresponding edges in G meet at a vertex. Each edge in $E(G)$ has an associated relative scale $\delta_{ijk} \sim \text{Log-}\mathcal{N}(d_{ijk}, g_{ijk}^2)$, and connects vertices corresponding to relative positions from i to j and j to k.

Edges in $E(G)$ are assigned weight g_{ijk}^2, then the 'shortest' path P between two nodes in $E(G)$ provides a sequence of relative scales from which the length of one relative position in terms of the length of the other relative position is estimated. This scale estimate has distribution $\text{Log-}\mathcal{N}(D, G^2)$ where $G^2 = \sum_{e \in P} g_e^2$ by Equation 8. Note that P is the sequence of edges minimising G^2. The following observations justify this weighting scheme:

1. If a scale $s_{ijk} \sim \text{Log-}\mathcal{N}(d_{ijk}, g_{ijk}^2)$ is parametrised from two point sets X and Y, then the scale computed in the opposite direction has the distribution $s_{kji} = \frac{1}{s_{ijk}} \sim \text{Log-}\mathcal{N}(-d_{ijk}, g_{ijk}^2)$. Therefore edges have the same weight regardless of the order or direction they are traversed[1].

2. If the measured relative scales were truly lognormal random variables $s_{ijk} \sim \text{Log-}\mathcal{N}(d_{ijk}, g_{ijk}^2)$, then P is the path giving an estimate of the log of the true scale with expected error less than the estimate along any other path.

[1]When D is calculated along a path in E(G), and rotations and translations are accumulated along a path in G the direction edges are traversed is relevant, this is implemented in an undirected graph simply by assigning each edge a relative pose/scale in one direction (the direction of increasing time) then reversing it if traversed in the opposite direction.

3. Gross errors in relative position and orientation normally lead to few accurately reconstructed points from which to recover scales, leading to higher g^2 estimates, by Equation 5. These edges will only be used when no better relative position estimates exist.

4. An alternative weighting scheme would be to weight each edge in $E(G)$ by the negative log of its inlier probability. The shortest path would then be the one least likely to contain an outlier, i.e. the most robust position estimate. These weights would be equal if $P(\text{scale estimate is an inlier}) = e^{-g^2}$. While the true outlier probability is hard to measure and is threshold-dependent in real data, coincidentally this equation does give realistic values of $P(\text{no outliers})$ of about 0.6-0.9 after travelling for 500 frames without loop-closure, and hence we believe it is a good strategy for minimising the probability of our position estimates being contaminated by outliers.

As at most a fixed number of relative positions is introduced to G at each time the number of edges in G, and hence the number of relative-scale-edges in $E(G)$, is proportional to the number of frames, T. Therefore a shortest path tree can be found in $E(G)$ in time $O(T \log T)$ using Dijkstra's algorithm. This shortest path tree gives a good position estimate for every frame and hence a global map. Alternatively a locally accurate map could be found in constant time by recursing to a fixed depth from the current node, in the same manner as Mei et al. (2009).

3.3.2 Motion models and scale-drift

While BoWSLAM can position a robot for extended periods without making assumptions about the robot's motion, in some visually challenging environments significant scale-drift accumulates and introduces errors into global maps (for example Figure 11(a)). This scale drift arises primarily from a small number of relative scale estimates where few 3D points are matched between successive frames; these unreliable scale estimates are incorporated into the map only when no better scale estimates have been found (usually when cornering rapidly). One solution to reduce this scale drift is to impose a weak motion model on relative scale estimates. Two possible motion models are proposed in this section. The first models the acceleration, and the second models the perceived depth of points in the world.

The first motion model is based on the assumption that the camera velocity is approximately constant over short periods of time. This motion model is implemented by assuming that the relative acceleration over short period of time is lognormally distributed about zero.

Relative scale estimates are modeled as lognormally distributed variables, which are parametrised by aligning reconstructed 3D points (Section 3.2.2). When a scale is computed from three frames, i, j and k, which are temporally close, we assume these relative scales are lognormally distributed about the scale corresponding to zero acceleration as follows:

$$\delta_{ijk} \sim \text{Log-}\mathcal{N}(log\frac{\Delta_{jk}}{\Delta_{ij}}, G_{MM}^2 \max(\Delta_{ij}, \Delta_{jk})) \tag{9}$$

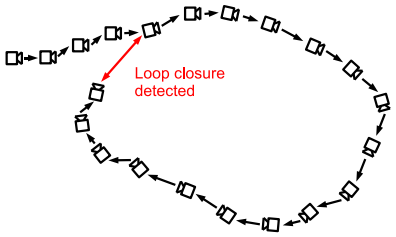
where Δ_{ij} is the time difference between frames i and j, and G_{MM} controls how tightly speeds are constrained. This distribution is combined with the distribution for δ_{ijk} from aligning points. G_{MM} is chosen so that when many points are observed, the scale from observing the points dominates, and when few are observed, the motion model dominates. This motion model is weaker than other single camera SLAM motion models as only the camera's relative acceleration is constrained. Even so, like other SLAM schemes, it could potentially limit camera acceleration after the camera has slowed. In this circumstance however, position estimates from many recent frames are normally available, as there is high overlap between consecutive frames when the camera moves slowly. If sufficient 3D structure is aligned over any of these triples of frames then this scale estimate will dominate.

This motion model reduces scale drift, but does not eliminate it; errors will still accumulate. An alternative motion model is based on the observation that reconstructed points in the world typically have depths in a particular range, for example 0.5m to 20m for a pedestrian outdoors. The mean depth of reconstructed points is proportional to the baseline length (which is assumed to be lognormally distributed), therefore by modelling the depth of reconstructed points with a lognormal distribution and combining this estimate with the depths from SLAM an absolute constraint on scales can be imposed. As sequences of scales are all dependent (by Equation 8), an uninformative distribution can be used, leading to many small updates that together eliminate scale drift over long sequences of frames. This

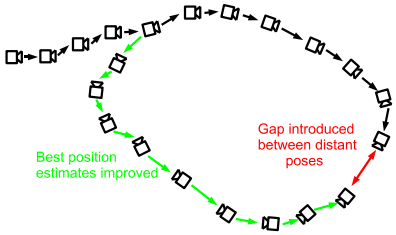
constraint is similar to constraints on the depth of observed points imposed by some other single camera SLAM schemes (Davison, 2003; Lemaire and Lacroix, 2007).

3.4 Loop closure and map optimisation

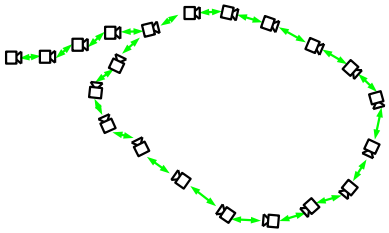
Loop closure in a graph-based map is very simple—an edge is added in the same way as for other relative position hypothesis. However when a loop is closed and a new global map is computed a break in a continuous chain of positions in the middle of the loop often forms when poses for frames at either side of the break are computed via different paths (Figure 9). This break will usually be far from the root edge from which the graph's shortest path tree is generated so a locally accurate map can always be constructed by setting the root near to the current robot position, however sometimes a global map is required.



(a) Loop closure detected



(b) Position estimates updated



(c) t-spanner optimised

Figure 9: When loop closure is detected the latest frame can be positioned relative to a much earlier frame (a). After loop-closure, the 'best' position estimate for each frame is updated. This improves the position estimates of a sequence of nearby frames relative to the latest frame (b). When a global map is required a t-spanner of good position estimates preserving cycles is selected and optimised, closing any distant gaps (c).

Our goal is to compute an accurate map from a set of relative pose estimates. Several solutions have been proposed for the case when errors in relative poses are Gaussian and independent (Section 2.1). We use the 3D version of the TORO framework (Grisetti et al., 2007a) to generate global maps.

The TORO framework is based on on earlier work by Olson et al. (2006). Olson et al. formulate the SLAM problem as a network of robot poses connected by uncertain relative pose estimates, then find the robot poses minimising the total residual in relative positions. Solving this problem via global methods such as Levenberg-Marquardt nonlinear optimisation is not computationally feasible, and linearising the problem and solving via a Kalman Filter can lead to significant artefacts from linearisation. Instead an approximate nonlinear solver based on Stochastic Gradient Descent is used. This method starts from an approximate initial solution, selects one constraint (relative pose estimate), calculates the adjustment to the robot poses needed to reduce the error in this constraint, then applies this adjustment to each robot pose. Instead of choosing constraints at random, as in Stochastic Gradient Descent, each is considered in turn. This, together with the structure of the problem (updates usually only affect temporally close/subsequent positions) means that updates can be applied to a tree of stored updates, so that sequences of successive nodes can be updated efficiently without having to iterate over them. This update is repeated for each constraint on each iteration. Updates are increasingly damped on each successive iteration, and iterations continue until the total residual is low. The effect of each iteration is

to partly close gaps in loops, distributing the error around the loop.

In the TORO framework (Grisetti et al., 2007b) this method is improved by building an update tree based on spatial rather than temporal proximity of nodes. This reduces the number of nodes that must be updated in most environments, particularly where a robot repeatedly visits the same area.

These approaches work in 2D, however they do not easily generalise to 3D as 3D rotations are not commutative. Instead, for the 3D version of TORO (Grisetti et al., 2007a), the translational and rotational components of motion are separated and updated alternately. Errors in rotation are distributed over sequences of relative poses by spherical linear interpolation.

There are two issues with using TORO with BoWSLAM's maps however, the first is that TORO does not model outliers, and one bad relative pose estimate can corrupt the entire global map. Optimising the entire pose graph G rarely results in a map resembling the ground truth, so instead we must select an outlier-free spanning subgraph of G. The most important property of this subgraph is that large cycles in G should be preserved. A suitable subgraph with this property is known as a t-spanner, $S_t(G)$, defined as a spanning subgraph where two nodes separated by a distance k in G are separated by no more than tk in $S_t(G)$. In a weighted graph with e edges a t-spanner with low total cost (close to the minimum possible) can be found efficiently in time $O(e \log e)$ using the algorithm by Althöfer et al. (1993). A value of around 20 for t gives a subgraph consisting of the minimal spanning tree plus a small number of edges sufficient to preserve cycles larger than t; this is the subgraph we optimise using the TORO framework.

A second problem with using TORO for single camera SLAM is that scale drift means that the independence assumption between poses does not hold. TORO distorts graphs containing significant scale drift. We plan to modify TORO to incorporate a cyclic constraint on scales in the same way that rotations around cycles are currently handled: around a cycle C, $\sum_{(ijk)\in C} d_{ijk} = 0$. For now we impose this constraint via a Singular Value Decomposition (SVD), which has poor complexity, but in practice is fast (a few milliseconds) in t-spanners as long sequences of edges typically belong to the same set of cycles, so can be temporarily replaced with a single edge. Scale drift within cycles still leads to small distortions in optimised maps.

The final requirement for optimisation are variances for each relative pose (full covariances are not used by TORO in 3D but could be estimated in the same way). These are estimated by assuming pose estimation is locally linear, then fitting a simple model to errors in simulated noisy data. While many factors contribute to these errors, the number of inliers found, D (pose estimation has $D-(5-1)$ degrees of freedom), and the uncertainty in point localisation (σ_L, about 0.69 pixels for the FAST corner detector) account for most of the error, giving $\sigma = K\frac{\sigma_L}{\sqrt{D-4}}$ with $K \approx 28$ for errors in orientation, and $K \approx 40$ for errors in translation direction. Approximating the lognormally distributed scales with normal distributions and scaling the errors in translation appropriately gives the required variances.

Other authors (Konolige et al., 2009; Eade and Drummond, 2007) use the condition of the Jacobian matrix from fitting a transformation to inliers to estimate transformation quality; in BoWSLAM however errors in position and orientation are usually caused by remaining outlier correspondences, either from movement in dynamic environments, or when a poorly-conditioned or small inlier set is found. We have found the Jacobian is often better conditioned when more outliers are present, so is a poor measure of the quality of a transformation.

4 Experimental results

In this section we demonstrate that BoWSLAM can robustly navigate difficult environments with three datasets, each processed off-line on a computer with a 3GHz Intel Core 2 Duo processor.

The first dataset is captured at 2.8Hz from a handheld camera in a suburban environment (Figure 10). Images are greyscaled, normalised, and down-sampled to 640×426. After travelling 1.6km the camera is stopped, then restarted elsewhere. A couple of minutes later the new path re-joins the original path. The sequence includes many pedestrians and cars, and three frames show only of the side of a passing bus.

Without assuming any motion model BoWSLAM successfully positions the robot along the first path, producing locally consistent maps (Figure 12a). Loop closure is detected every time the path re-visits an earlier location. When the camera is stopped and re-started a new map component is started (Figure 12b), however significant scale drift accumulates during rapid cornering. When the new path returns to a previously visited location loop-closure is detected and the maps are fused. The scale from

Figure 10: Frames from the Suburban dataset, covering 2.5km over 25 minutes. Dozens of pedestrians and cars pass by, and one sequence of three frames is completely obliterated by a passing bus.

the first component is propagated along the new path, however a loop-closure opportunity was missed when the camera travels in the opposite direction to the previous visit, leading to large uncorrected scale drift, as seen on the right of Figure 12c). This loop is hard to detect as features common to both frames are visible from very different angles; even viewpoint-invariant SURF descriptors are not sufficient to detect this loop closure.

Initially a framerate of 16Hz is maintained; after 24 minutes this has dropped to 11Hz; still almost four times the rate images were captured. Correcting scale drift in 1546 edges, around 7 cycles, by SVD takes just 60 milliseconds.

To produce the optimal map in Figure 11 a 25-spanner of the pose graph is found. This contains 14 more edges than the shortest path tree used to find locally consistent maps (Figure 12). A reasonable map with only small gaps is obtained after 500 iterations (6 seconds). After 20 000 iterations (234 seconds) the global error is reduced four-fold and gaps are completely closed. This slow convergence is due to variation in relative position variances; by setting all variances equal a reasonable map is obtained in 50 iterations. While most details of the ground-truth track from GPS are recreated, in two places (at either end of the second track) uncorrected scale-drift corrupts the position estimate.

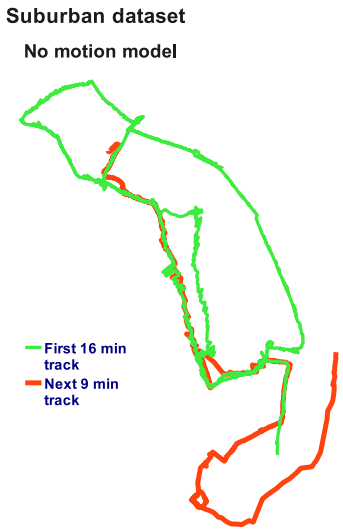
Table 1: RMS errors in global maps optimised by TORO. Errors are relative to GPS positions after scaling and aligning maps; RMS errors in the non-differential GPS position are estimated to be 5m. These figures are parametrisation-dependent and mainly reflect the amount of scale-drift present. The total track length is 2.5km.

Motion model	RMS error
No motion model	198m
Constrained acceleration	83m
Constrained depths	56m

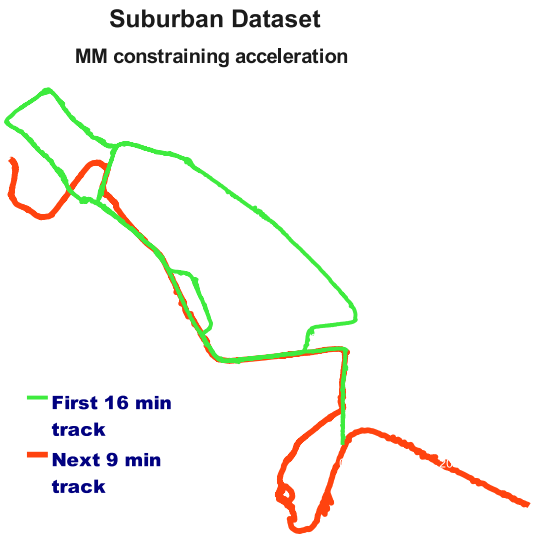
We have also processed this dataset with the two motion models proposed in Section 3.3.2. The first of these motion models, constraining accelerations, reduces scale drift throughout the map (Figure 11(b)), however some scale drift still distorts the map during tight, rapid cornering. The constraint on acceleration does not affect BoWSLAM's ability to stop or start moving at the three locations where the camera is stationary.

The second motion model, constraining the depths of reconstructed points, greatly reduces scale drift and all features of the original map are accurately reconstructed, however some distortions are introduced where the camera passes through a narrow passage where the point depth model does not reflect the true point depths (Figure 11(c)). The RMS errors between these three runs and the GPS ground-truth are given in Table 1, showing that both motion models improve BoWSLAM's accuracy.

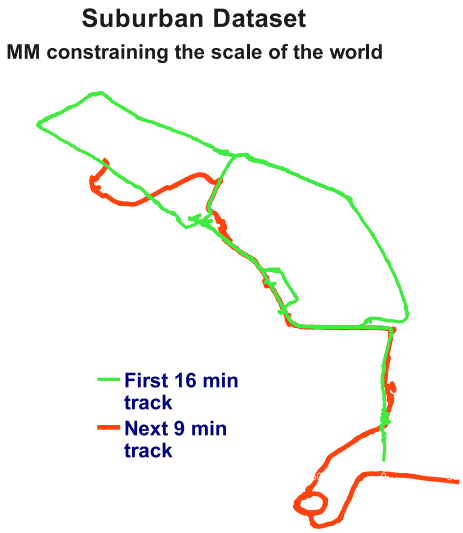
The second dataset demonstrates BoWSLAM navigating a typical indoor environment. The dataset, from the University of Alberta (The Robotics Data Set Repository (Radish), 2003), consists of a 75m



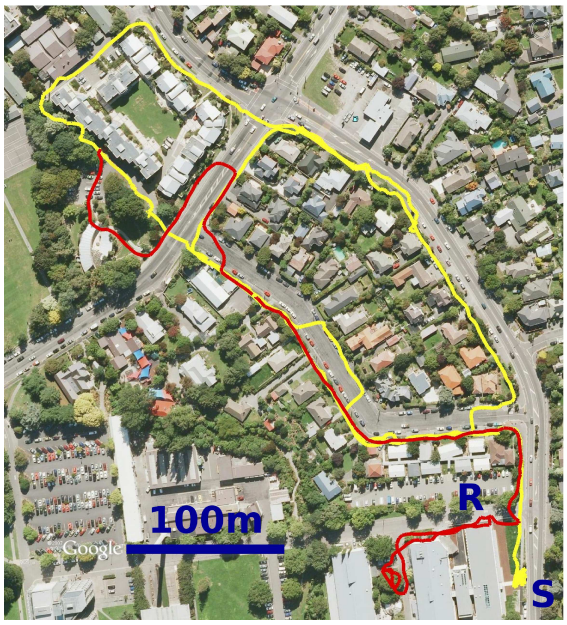
(a) Global map, no motion model. Most features of the path are reconstructed accurately, except at either end of the second trajectory where loop-closure events are missed and scale-drift accumulates.



(b) Motion model constraining acceleration. Many artifacts eliminated, although scale drift still occurs at the start of the second track.

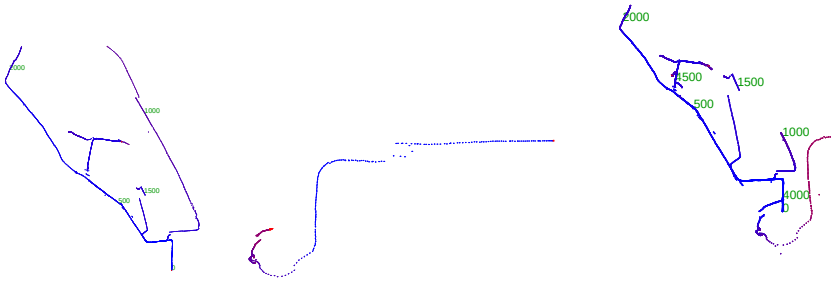


(c) Motion model constraining allowed scale of the world. Scale drift is greatly reduced, although the map is distorted where the true scale of the environment does not reflect the depth distribution assumed by the motion model.



(d) Ground truth data from GPS

Figure 11: Maps of robot poses compared with ground truth from GPS, from the Suburban dataset (Figure 10). The path starts at **S** and traverses the large loop twice, with various diversions. After 16 minutes the camera is stopped, then restarted at **R**. Later the original path is re-joined.



(a) A locally-consistent map is built as the robot explores.

(b) The camera is stopped after 16 minutes and started elsewhere. A new map component is started (rotated to match Figure 11).

(c) Loop closure is detected as soon as path crosses previous path and the two map components are fused.

Figure 12: The robot needs to identify which frames have reliable position estimates as it explores. To do this it maintains maps where each frame is positioned along the 'shortest' path back to a root node (the first node in this example). These maps are locally consistent (near the root) but contain significant scale drift and gaps in loops; these are corrected when a globally accurate map is optimised using TORO.

Figure 13: Frames from the University of Alberta CS Centre dataset—a 512 frames captured around a 75m indoor loop from The Robotics Data Set Repository (Radish) (2003).

rectangular loop around corridors and landings. 640×480 greyscale images were captured every 15cm from a wheeled robot (Figure 13). Images are processed at 16Hz, corresponding to a robot velocity of 2.4m/s. Positioning is accurate along straight corridors, despite a large number of self-similar features, and features reflected from the uneven floor. Again when cornering rapidly the rapid rotation relative to forward motion make tracking scale difficult, as few points with accurate 3D positions are tracked into subsequent frames (using a wide-angle lens would greatly reduce this problem). Positions of a few frames with gross errors are visible near corners; these errors are successfully avoided when computing subsequent positions.

When the start point is revisited loop closure is detected.Optimisation with the TORO framework produces the map shown in Figure 14, needing 50 iterations and 0.08 seconds. Artefacts from the errors during rapid cornering are still visible, in particular the zig-zagging where a sequence of alternate frames is positioned relative to two different earlier frames.

The final dataset demonstrates BoWSLAM navigating a challenging outdoor dataset. The dataset is captured from a Contour HD mountain bike helmet camera, and features full six degree-of-freedom motion, large changes in scale, rapid cornering, erratic motion over rough ground, and sequences of frames with motion blur (Figure 15). 1.5km are covered in 5 minutes at speeds ranging from 5 to 30 km/h. The original video consists of frames sized 1280×720 captured at 30Hz; image features accelerate from zero to 140 pixels-per-frame across sequences of three frames, and sequences of up-to six consecutive frames contain motion blur.

One-in-three frames from the original video are used, downsampled to 640×360. A motion model constraining the world's scale, and scale- and rotation-invariant SURF descriptors are used; these are

necessary to register poor quality images despite rapid changes in scale, but cut performance to around two frames-per-second.

BoWSLAM successfully recreates the double-loop, and many features of the trajectory, despite numerous small errors (Figure 16). Overall the map is warped from many small errors in relative orientation, and errors in scale when the camera moves between forest and open areas. Corrections around loops mean changes in altitude are recreated successfully, despite the warped map (Figure 17).

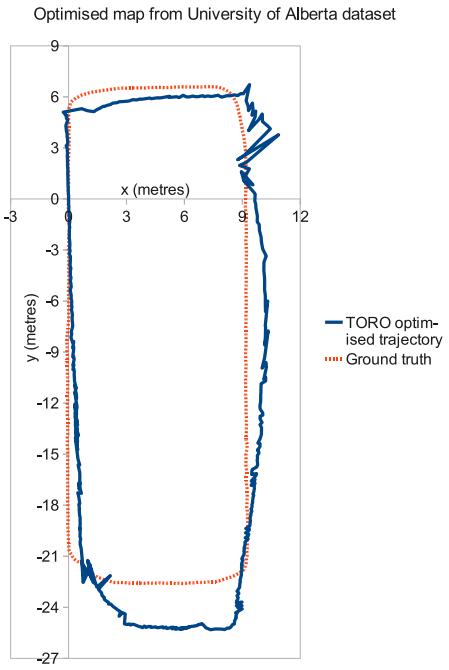


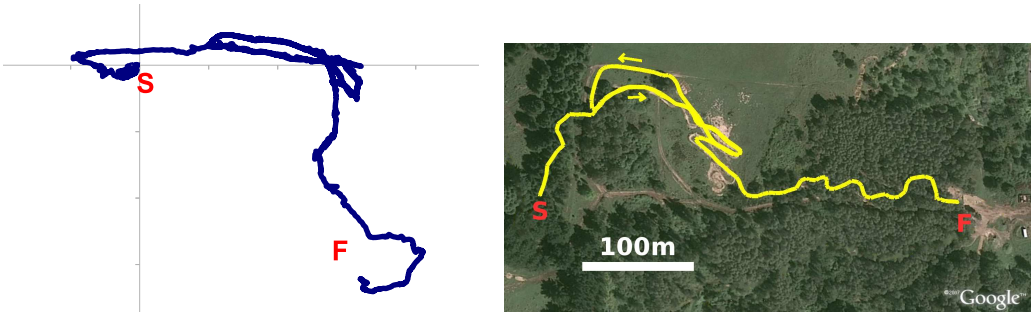
Figure 14: Global map of robot poses compared with ground-truth, University of Alberta dataset (Figure 13), optimised with TORO. A scale is applied to match the ground-truth.

Figure 15: Frames from the Helmet Camera dataset, a challenging sequence characterised by erratic motion and motion blur.

4.1 Long-term real-time operation

A breakdown of typical processing times is given in Figure 18. Note that once the inlier correspondences and epipolar geometry are determined, the positioning and mapping parts of the algorithm are very fast. Also computing four position hypotheses per frame rather than one only increases the total cost by about 50%. For this dataset, using RANSAC instead of BaySAC almost doubles the time needed to find essential matrices (increasing the total time by 20%), although this is parametrisation-dependent.

For the first few thousand frames the execution time is dominated by constant-time operations: extracting features from each frame and computing positions relative to around four nearby frames. However, the following parts of BoWSLAM currently have higher complexity in the number of frames T, and will eventually become the most expensive operations:



(a) BoWSLAM's optimised map of the Helmet Camera dataset.

(b) Ground-truth data from GPS.

Figure 16: Optimised map of the Helmet Camera dataset. BoWSLAM successfully re-creates the structure of the trajectory, although errors accumulate around tight corners and distort the optimised map.

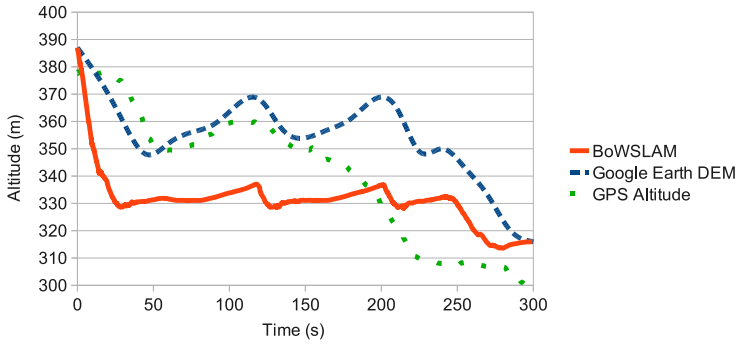


Figure 17: Camera altitude on the Helmet Camera dataset from GPS, from Google Earth's Digital Elevation Model (DEM), and from BoWSLAM (shifted and scaled to match the range of the DEM data). BoWSLAM's altitude estimate contains systematic errors from a slightly warped map, however details are recreated more successfully than by GPS, where the same location has a 30m height difference on two visits (although differential GPS would eliminate this error).

- Loop closure takes time up to $O(T \log T)$ (Section 3.3)
- Finding BoW matches takes time $O(T)$ (comparison with every other frame)
- Building a new BoW dictionary (clustering) takes time $O(T \log T)$, although per-frame this is only $O(\log T)$
- Finding a t-spanner for optimisation by TORO takes time $O(T \log T)$, and TORO takes time $\Omega(T)$, although this is performed only occasionally when a globally-accurate map is needed.

For long-term operation, constant time costs are necessary, however in a bounded environment a robot will eventually map everywhere it ventures, reducing the SLAM problem to a much simpler localisation problem. In the mean time the costs listed here could still be reduced further: BoW matching can be speeded up (to $O(\log T)$) by clustering the BoW image representations and only searching in nearby clusters (Fraundorfer et al., 2007). BoW clustering can be avoided entirely once a dictionary appropriate for the environment is found; alternatively for some environments a 'general purpose' dictionary (Cummins and Newman, 2008a), could be used, avoiding the need to cluster entirely.

Memory usage is linear in the number of frames and is dominated by storing descriptors, requiring around 40KB per frame. This gives about 25 000 frames/two hours of operation per GB of memory. This could be reduced greatly if correspondences directly from the BoW representation were used (as proposed in Section 3.1.1)—this would remove the need to store descriptors once a BoW dictionary representative of all the environments that might be encountered was found.

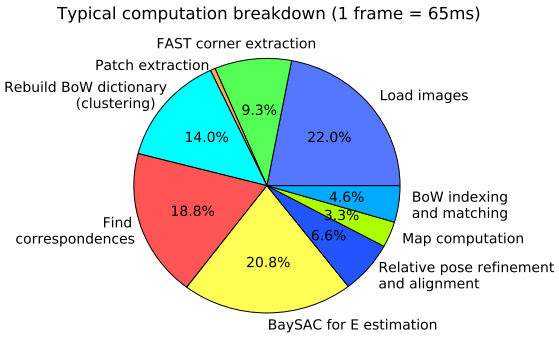


Figure 18: Breakdown of execution time per frame. Computed on University of Alberta dataset, excludes rendering of maps and TORO optimisation.

4.2 Discussion

BoWSLAM can navigate in three dimensions with very few assumptions about motion, however a motion model helps to reduce scale drift. Ideally rotation-invariant descriptors should be used, as some robots (for example climbing robots) may visit the same location with a different orientation. Rotation-invariant SURF descriptors perform as well, or better than patch descriptors, however they are currently too costly to extract in real-time.

We attempted to parametrise MonoSLAM (Davison, 2003) to work on the University of Alberta dataset, however features near to the camera move too far between frames to be tracked (or when allowed motion is increased the tracking jumps between nearby similar-looking features). When a corner is reached, rapid feature motion causes tracking to fail, and the estimated path continues straight ahead due to the constant velocity motion model.

Like all SLAM schemes with global loop-closure detection, there is a risk that BoWSLAM may close loops incorrectly in self-similar environments, corrupting part of the map, however this is only a problem if the geometry of the scenes is also compatible. So far the only incorrect loop closures observed have been in simulated data, where the same image is observed in different locations. Other schemes (Cummins and Newman, 2009; Konolige et al., 2009) have also found geometric constraints sufficient to prevent false loop closure. In addition, if incorrect loop closure could be detected the damage could be undone simply by deleting the incorrect link.

5 Conclusions

This paper describes BoWSLAM, a scheme enabling real-time navigation of dynamic environments using a single camera alone. An accurate map is generated after travelling 2.5km over 25 minutes; considerably further than the tracks over which previous single camera SLAM schemes have been demonstrated. BoWSLAM demonstrates that autonomous mobile robots may be positioned for extended periods in large-scale environments using just a single camera.

Three innovations make this possible. Firstly, the use of the BoW algorithm, together with BaySAC, for fast, wide-baseline feature matching. This enables positioning despite substantial camera movement and outliers from moving or repeated features, and is necessary when computing position estimates relative to more distant frames. Secondly, the BoW algorithm is used to select good candidate frames from which to compute positions, this allows positioning to continue despite sequences of frames being unusable, for example due to moving objects. Thirdly, a graph-based representation of multiple position hypotheses allows subsets of good position estimates to be found, despite the presence of gross outliers. Scale estimates are modeled as a dual graph, which is optimised to correct scale drift when loops are closed.

6 Future development

We have demonstrated BoWSLAM's robustness to gross errors and challenging environments, however there are several areas where accuracy can be improved. Firstly, scale drift is still a large source of errors in maps, particularly when navigating by dead-reckoning (far from previously-visited locations). One obvious solution is to incorporate measurements from additional sensors; in particular the complementary measurements from low-cost IMUs (like those found in many modern phones). An alternative solution however, on which we have carried out some preliminary work (Botterill et al., 2009a), is to extend the high-level representation of each frame by using the BoW database to learn and recognise classes of objects. Measurements of the size of these objects can then be used to estimate the true scale of the world.

We also plan to improve positioning accuracy by using a bundle adjustment-based approach (Triggs et al., 1999; Mouragnon et al., 2009). Position estimates over good sequences of frames could be refined, and information from multiple good tracks could be combined (Section 3.3), however a better approach may be to focus on those regions with the highest uncertainty.

7 Acknowledgements

The University of Alberta CSC data set was obtained from The Robotics Data Set Repository (Radish) (2003). Thanks go to Jonathan Klippenstein for providing this data.

References

I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Journal of Discrete and Computational Geometry*, 9(1):81–100, 1993. ISSN 0179-5376. doi: http://dx.doi.org/10.1007/BF02189308.

A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. Real-time visual loop-closure detection. In *Proceedings of the International Conference on Robotics and Automation*, 2008a.

A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, Special issue on Visual SLAM: 1–11, 2008b.

V. K. Balakrishnan. *Schaum's outline of theory and problems of graph theory*. McGraw-Hill, 1997.

H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110:346–359, 2008.

J. S. Beis and D. G. Lowe. Indexing without invariants in 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1000–1015, 1999.

M. Bosse, P. Newman, J. Leonard, and S. Teller. Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. *International Journal of Robotics Research*, 23: 1113–1139, 2004.

T. Botterill, S. Mills, and R. Green. Speeded-up Bag-of-Words algorithm for robot localisation through scene recognition. In *Proceedings of Image and Vision Computing New Zealand*, pages 1–6, Nov. 2008. doi: 10.1109/IVCNZ.2008.4762067.

T. Botterill, R. Green, and S. Mills. A Bag-of-Words Speedometer for Single Camera SLAM. In *Proceedings of Image and Vision Computing New Zealand*, pages 1–6, Wellington, NZ, November 2009a.

T. Botterill, S. Mills, and R. Green. New conditional sampling strategies for speeded-up RANSAC. In *Proceedings of the British Machine Vision Conference*, 2009b.

L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardos. Mapping large loops with a single hand-held camera. In *Proceedings of Robotics: Science and Systems*, 2007.

G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proceedings of the ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

M. Cummins and P. Newman. Accelerated appearance-only SLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 1828–1833, May 2008a. doi: 10.1109/ROBOT.2008.4543473.

M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *International Journal of Robotics Research*, 27(6):647–665, 2008b. doi: 10.1177/0278364908090961.

M. Cummins and P. Newman. Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2003.

F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, 25:1181–1203, 2006.

G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In *Proceedings of the International Conference on Robotics and Automation*, pages 1009–1014 vol.2, 2000. doi: 10.1109/ROBOT.2000.844732.

E. Eade. *Monocular Simultaneous Localisation and Mapping*. PhD thesis, University of Cambridge, 2008.

E. Eade and T. Drummond. Scalable monocular SLAM. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 469–476, Los Alamitos, CA, USA, 2006. IEEE Computer Society. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2006.263.

E. Eade and T. Drummond. Monocular SLAM as a graph of coalesced observations. In *Proceedings of the IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, October 2007.

E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular SLAM. In *Proceedings of the British Machine Vision Conference*, 2008.

C. Estrada, J. Neira, and J. D. Tards. Hierarchical slam: realtime accurate mapping of large environments. *IEEE Transactions on Robotics*, 21:588–596, 2005.

D. Filliat. A visual bag of words method for interactive qualitative localization and mapping. In *Proceedings of the International Conference on Robotics and Automation*, 2007.

M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. ISSN 0001-0782.

F. Fraundorfer, H. Stewénius, and D. Nistér. A binning scheme for fast hard drive based image search. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2007.

U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196–207, April 2005. ISSN 1552-3098. doi: 10.1109/TRO.2004.839220.

P. Gemeiner, A. J. Davison, and M. Vincze. Improving localization robustness in monocular slam using a high-speed camera. In *Proceedings of Robotics: Science and Systems*, 2008.

G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007a.

G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proceedings of the Robotics: Science and Systems (RSS)*, 2007b.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, second edition, 2003. ISBN 0-521-54051-8.

K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

T. Lemaire and S. Lacroix. SLAM with panoramic vision. *Journal of Field Robotics*, 24:91 – 111, 2007.

F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1150–1157, 1999.

C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. A constant-time efficient stereo SLAM system. In *Proceedings of the British Machine Vision Conference*, 2009.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002.

J. M. M. Montiel, J. Civera, and A. J. Davison. Unified inverse depth parametrization for monocular SLAM. In *Proceedings of Robotics: Science and Systems, Philadelphia, USA*, 2006.

E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 27(8):1178–1193, 2009. ISSN 0262-8856. doi: http://dx.doi.org/10.1016/j.imavis.2008.11.006.

P. Newman, D. Cole, and K. Ho. Outdoor SLAM using visual appearance and laser ranging. In *Proceedings of the International Conference on Robotics and Automation*, Florida, 2006.

P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schrter, L. Murphy, W. Churchill, D. Cole, and I. Reid. Navigating, recognising and describing urban spaces with vision and laser. *International Journal of Robotics Research*, 28:1406–1433, 2009.

T. Nicosevici and R. Garca. On-line visual vocabularies for robot navigation and mapping. In *Proceedings of IROS*, pages 205–212, 2009.

D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, June 2006.

D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

NIST/SEMATECH e-Handbook of Statistical Methods, March 2010. Carroll Croarkin and Paul Tobias, from http://www.itl.nist.gov/div898/handbook/.

E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2262–2269, 2006.

E. Rosten and T. Drummond. Machine leaning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision*, pages 430–443, 2006.

P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*. John Wiley & Sons, Inc., New York, NY, USA, 1987. ISBN 0-471-85233-3.

J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1470–1477, Oct. 2003.

R. Smith, M. Self, and P. Cheeseman. *Autonomous Robot Vehicles*, chapter Estimating Uncertain Spatial Relationships in Robotics, pages 435–461. Springer Verlag, Amsterdam, 1990.

H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.

H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale drift-aware large scale monocular slam. In *Proceedings of Robotics: Science and Systems Conference*, 2010.

The Robotics Data Set Repository (Radish), 2003. Andrew Howard and Nicholas Roy, from http://radish.sourceforge.net/.

H. C. Thode. *Testing for normality*. CRC Press, 2002.

S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal of Robotics Research*, 25(5-6):403–429, 2006.

B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, volume 1883/2000, pages 1–71, Corfu, Greece, 1999.

B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardos. An image-to-map loop closing method for monocular SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.