Scalable Localisation and Mapping of Textured Scenes

Author:
Jai Juneja
Balliol College

Supervisor:
Dr Andrea Vedaldi
Visual Geometry Group

A thesis submitted for the degree of

Master of Engineering in Engineering, Economics & Management

Trinity Term 2014

Abstract

This report develops a large-scale, offline localisation and mapping scheme for textured surfaces using the bag-of-visual-words model. The proposed system builds heavily on the work of Philbin et al. [19] and Vedaldi and Fulkerson [29], taking as an input a corpus of images that are rapidly indexed and correlated to construct a coherent map of the imaged scene. Thereafter, new images can be reliably queried against the map to obtain their corresponding 3D camera pose. A simple bundle adjustment algorithm is formulated to enforce global map consistency, exhibiting good performance on real datasets and large loop closures. Furthermore, a proposed submapping scheme provides dramatic computational improvements over the baseline without any deterioration in localisation performance. A comprehensive implementation written in MATLAB as proof of concept is pressure tested against a variety of textures to examine the conditions for system failure. The application unambiguously outperforms the naked eye, demonstrating an ability to discriminate between very fine details in diverse settings. Tests validate its robustness against broad changes in lighting and perspective, as well as its notable resilience to high levels of noise. Areas for further development to augment the original work and achieve real-time performance are also suggested.

Acknowledgements

I would like to thank my supervisor Dr Andrea Vedaldi, whose constant support and enthusiasm over the past year has made my project experience a delight.

Contents

Chapter 1: Introduction

1.1 Motivation and Problem Statement

In recent years, computer vision techniques have become central to the fields of robotics and artificial intelligence. Cameras serve as a cheap and information-rich alternative to more traditional sensors such as laser rangefinders. Not only has this enabled us to explore new avenues in localisation, mapping and object identification, but also to exploit the unparalleled richness of information that exists in images.

Decades ago, many scientists would have rendered high-performance machine vision an impossible task. Indeed, human vision is by comparison a highly evolved and sophisticated sensor modality supported by several hundred million neurons in the visual cortex. One must therefore concede that achieving general purpose computer vision remains a distant, if not impossible, challenge. However, modern systems have adopted many ideas from neurophysiology and are capable of performing well-defined, specialised tasks such as accurate 3D reconstruction better than humans. This project focuses on another such task: unlocking useful information from textured photographs of planar scenes, which to the naked eye appear uninformative or homogeneous. Planes are commonly found in urban, indoor and, at some scales, even natural settings. Furthermore, whilst there has been growing interest in the field of vision-based localisation and mapping, relatively little has been done to develop algorithms that are robust in textured environments. This project aims to build a framework to coherently interpret and reconstruct these scenes, thus allowing any query image to be rapidly and reliably localised within it. Such technology has a wide range of potential applications, including:

- Tracking and localisation of aeroplanes and unmanned aerial vehicles mounted with downward-facing cameras. This could include identification, tagging and mapping of landmarks on the ground.
- Satellite photogrammetry used to build large-scale maps of entire cities. Landmarks or query images captured from overhead could then be rapidly located.
- Offline construction of floor maps for domestic or industrial robots that are regularly rebooted, helping to solve the "kidnapped robot problem" as they can immediately query their position on startup.

In general the scenarios under consideration involve downward-facing cameras. This configuration has a number of benefits: for example, the scenes within view are unlikely to be occluded; the flat geometry simplifies much of the computational and mathematical complexity; a high level of localisation accuracy

Figure 1.1: Sample images from our dataset. Some scenes have visibly repeating structures, whereas others are relatively indiscernible to the human eye.

can be achieved if the camera is adequately close to the target surface; and importantly, new textural data is exploited, yielding information that can complement or replace non-vision based systems.

Rosenholtz [21] describes texture as a substance that is more compactly represented by its statistics than the configuration of its parts. Textures may contain repeating patterns that can be easily classified at an aggregate level. However, at a finer scale they are often comprised of more intricate instance-specific details. For example, each line or pattern on wood is unique and constitutes a "textural fingerprint" of sorts. This complexity is something to be exploited: the fingerprints, whose compositions stand out from the local statistics, encode important information about the unique structure of an image. By detecting and describing them, we gain crucial insight into how seemingly disparate parts of a textured scene overlap and fit together.

We can characterise images and their local configuration by extracting keypoint features. In this report, we use the Scale Invariant Feature Transform (SIFT) to generate local features, and feed these into a mapping framework to coherently reconstruct the global scene in 3D space. Figure 1.1 exhibits a selection of scenes and textures from our dataset. Note the variability in lighting and angle of the photographs used to test the robustness of the system.

The final implementation is divided into two modules, as illustrated in Figure 1.2. First, a set of images is passed through the stitching and reconstruction module, which maps the scene in 3D using information about pairwise matches between images and assumptions about the geometry of the environment. Second, a query image can be input into the localisation module, which searches the world map of features to determine the camera pose and the projection of the image onto the world plane.

1.2 Project Outline

The remainder of this chapter offers a brief review of the state-of-the-art in vision-based localisation and mapping. Key literature and recent innovations relevant to the project are discussed, along with a summary of the simultaneous localisation and mapping (SLAM) problem.

Chapter 2 (*Stitching and Reconstruction*) tackles stage 1 of Figure 1.2 by building a mathematical

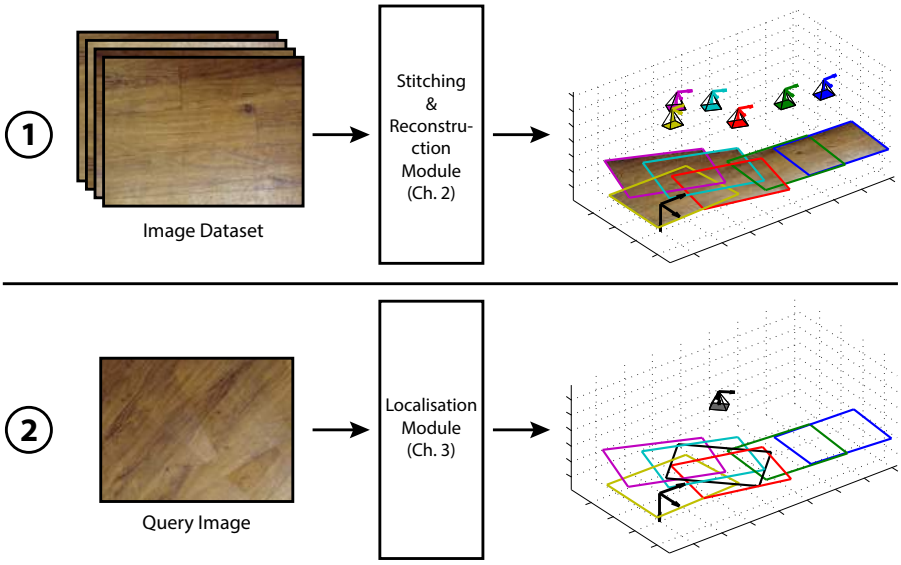


Figure 1.2: The world map is pre-computed using a database of images. Query images can then be passed through the localisation module, which returns a set of candidate camera poses.

framework to approach the problem. It begins by developing a simple system for 2D mapping based purely on image matching. This is subsequently extended to full 3D reconstruction and pose recovery by considering the geometric properties of the camera and world. A solution for bundle adjustment is also devised, such that feature positions and camera poses are globally optimised.

Chapter 3 (*Localisation*) details the formulation and implementation of the baseline localisation system used for rapid querying of images, as in stage 2 of Figure 1.2. An algorithm for generating submaps, shown to provide important improvements in system performance, is also proposed.

Chapter 4 (*Data and Evaluation*) begins by reviewing the corpus of images included in our dataset. The methodology for testing the localisation system is next described, followed by a detailed analysis of results.

Finally, Chapter 5 (*Conclusion*) reflects on the overall contributions of the report, providing a conclusion and evaluation of work carried out. A view on key areas for development going forward is also put forth.

1.3 Background

Motivations: At the outset, the question of why to employ a vision-based solution for large-scale localisation and mapping is important. One reason is that other solutions have numerous physical restrictions. For example, the Global Positioning System (GPS) typically exhibits errors of 2-10m, which may be too significant for safe navigation in populated or household environments. Furthermore, it is dependent on satellite signals for adequate operation, which can result in especially poor performance indoors or underwater, where satellites are often obscured. Laser rangefinders and ultrasonic

transceivers mounted on robots typically rely on nearby obstacles, such that time-of-flight distance measurements can be made using reflected pulses. By contrast, visual systems can potentially thrive in environments that lack physical obstructions or objects, but are rife with textural information. Vision is ideal for detecting loop closures as it captures rich and distinctive environmental data, thus ensuring that accumulated errors are corrected. This is critical for accurate long-term mapping and for localising a lost robot.

The low cost and accessibility of cameras has made them viable components for both household and industrial applications. This has helped to engender an active open source community, reinforced by the release of high quality open source libraries such as OpenCV (C++) and VLFeat (C/MATLAB), which provide a platform for experimentation and continuous improvement. Our implementation utilises the Visual Lab Features (VLFeat) library by Vedaldi and Fulkerson [29] as a source of popular computer vision algorithms.

Feature Extraction and Matching: The work presented in this report builds on a vast body of research in image matching, retrieval and mapping. Visual localisation relies on the ability of a query image to be matched against a target database or world map. To achieve this, we must establish a matching criterion by which images can be characterised and their similarity measured. Many such representations exist [14, 25, 19], such as colour histograms (which are compared via histogram intersection), feature points (edges, corners or blobs) and 'bag-of-visual-words'.

Feature extraction involves locating salient points in an image and assigning to each a *descriptor* vector, which encodes information about the local region such as intensity values or gradients. Features are then matched against a target database, typically based on the Euclidean (L_2) distance of their descriptor vectors. Good matching therefore requires highly distinctive descriptors that are unaffected by imaging conditions. The Scale Invariant Feature Transform (SIFT) proposed by Lowe [14] has been selected for its robustness to changes in image scale, orientation, illumination, affine distortion and 3D camera viewpoint. It is further detailed in Section 2.1.1.

Indexing and Retrieval: Although efficient extraction is a noted strength of SIFT descriptors, each keypoint vector is $\mathbb{R}^{128}$, making it cumbersome for large-scale storage and retrieval. The state of the art for image retrieval instead uses the bag-of-words (BoW) model. Sivic and Zisserman [25] disrupted this area by developing an object retrieval system based on the same principles as Google Search. In a close analogy to established methods for text document retrieval, image feature descriptors are vector quantised into visual 'words' using k-means clustering. Words are then weighted according to a *term frequency-inverse document frequency* (*tf-idf*) scheme, which takes the product of two terms:

the word frequency, which promotes words that are common in a particular document (and hence describe it well), and the inverse document frequency, which down-weighs words that appear often in the database. Each image is then described by a vector of weighted word frequencies, and at retrieval they are ranked according to their L_2 distance with the query vector.

Quantisation of SIFT descriptors enables us to assign a single integer value to each 128-dimensional feature vector, and compactly index them into an inverted document to achieve scalability. The inverted list structure contains an entry for each visual term followed by a list of all images in which it occurs. It has been shown to work well on datasets of 1 million images [19], allowing fast lookup of candidate images. However, the time to train a vocabulary by k-means can be extremely long due to the complexity of calculating nearest neighbours between points and cluster centres. Thus, a number of improvements to the system in [25] have since been devised.

One improvement proposed by Nistér and Stewenius [18] is to employ a more computationally efficient hierarchical k-means (HKM) algorithm to build a "vocabulary tree". Here, k defines a branching factor as opposed to the final number of cluster centres. Initially, the k-means process is run on the training data for one iteration. The data is then partitioned into k groups and the same process is applied recursively to each group, thus building the tree level-by-level. This reduces the time complexity of a single k-means iteration from $O(nk)$ to $O(n \log k)$, where n is the number of features used for clustering. Philbin et al. [19] alternatively examine an approximate k-means (AKM) algorithm, which addresses the main bottleneck: nearest neighbour computation. This replaces its exact calculation with an approximate nearest neighbour method, using a forest of randomised kd-trees. They demonstrate that, whilst this approach has the same time complexity as HKM, it provides improvements in recognition. Overall, these clustering improvements have enabled scaling up to much larger vocabularies in the order of 1 million visual words. The system described in [19] forms an important basis for the proposed localisation system and is revisited in Section 2.1.

A second improvement addresses the lack of spatial structure in the BoW representation. Philbin et al. [19] re-rank the retrieved list of image matches by estimating affine transformations between the query and target images and checking whether they are geometrically consistent. This is achieved using the RANSAC algorithm, which calculates several candidate transformations using sets of randomly selected feature correspondences. For each transformation, the number of "inlier" features is counted and the best score contributes to the new rank of an image. This geometric information is also crucial for image stitching, as described in Chapter 2. More comprehensive summaries of 3D reconstruction and bundle adjustment, which build on the initial 2D stitching, are provided in Sections 2.3 and 2.4.

Quantisation impacts the quality of matches and, in the context of localisation, raises the probability of a queried feature yielding multiple ambiguous matches in the world map. As the average number

of ambiguous matches rises, spatial verification becomes increasingly cumbersome. In Chapter 3, an improvement to this problem is offered by clustering the global map into appropriately-sized submaps.

1.3.1 BoW-based Monocular SLAM

Once the vocabulary tree is determined, new images can be queried and inserted into the database on-the-fly, at approximately the same rate as feature extraction for a single image [18]. Tests by Fraundorfer et al. [12] show that querying a database of 1 million images (with 200 words per image on average) takes 0.02s, giving a frame-rate of approximately 50Hz. However, SIFT features introduce additional computational burden that tends to reduce this rate to 1-5Hz [6]. Such a high retrieval performance is especially useful for vision-based simultaneous localisation and mapping (SLAM), where new locations need to be added incrementally in real time [18]. This is an exciting and highly scalable solution to SLAM. Once an initial map has been constructed using representative training images and a sufficient number of visual words, it can be rapidly augmented online. Pre-computation may be undesirable for some applications and as such alternative schemes exist to dynamically add words to a dictionary [2, 11]. However, real time performance for these is limited to a few thousand words – insufficient for long-term navigation.

SLAM is the problem in which a mobile robot must incrementally build a map of its environment whilst simultaneously determining its location within it. Individually, localisation and mapping can be formulated as estimation problems that are greatly simplified when the robot pose or map are known *a priori*. However, the difficulty of SLAM is to overcome the strong coupling between these two estimation tasks: without a coherent map, a robot cannot reliably determine its position; without good pose estimation, it cannot build a consistent map. Three additional challenges for SLAM are scalability (computational complexity), data association (matching consecutive measurements of the same landmark) and loop closure (recognising previously visited locations).

Whilst this project is not concerned with developing a comprehensive online solution for SLAM, the potential for extending the baseline BoW-based system for real-time applications should be assessed. In this context, and given the objectives of the project, the proposed system should meet a number of specifications such that it can be adapted for real time usage:

1. **Texture:** Robustly estimate position in self-similar (planar) environments – a common characteristic of textured scenes.

2. **Matching/Data Association:** Compute positions of frames captured from varying viewpoints; be robust to occlusions, lighting and camera distortions such as noise.

3. **Loop Closure:** Actively identify when known locations are re-visited and incorporate this

information into a global map. This is crucial for coherent mapping over extended periods.

4. **Optimisation:** Reject erroneous estimates that have the potential to corrupt a global map, and globally correct the map such that historical data are consistent.

5. **Scalability:** Operate smoothly over large environments.

Most contemporary SLAM solutions come in one of two forms [6]. The first calculates a maximum likelihood estimate (MLE) for the map and robot pose, whilst typically assuming that measurement and odometry errors are approximately normally distributed. The most widespread example is Extended Kalman Filter (EKF) SLAM, which uses a probabilistic motion model to predict the position of the robot and known landmarks, and subsequently uses a measurement model to correct these estimates in light of new sensory data [26]. This process is applied recursively, with the robot predicting, updating and augmenting its state vector and covariance matrix at each iteration. However, the complexity of correction equations is quadratic in the number of landmarks, as a large covariance matrix maintaining every landmark inter-relationship in the map must be continuously updated. This poses scalability and data association issues in unbounded environments where the number of features can grow indefinitely. The first and most popular scheme for single camera SLAM using the EKF was MonoSLAM [10]. Clemente et al. [8] improved the scalability of the original method by applying MonoSLAM to smaller submaps and optimising the transformations between these locally-accurate submaps to give a coherent global map.

The second SLAM formulation does not make any restrictive assumptions about world dynamics or the density function, but makes the observation that estimated landmark positions are independent given the robot's trajectory [6]. For example, particle filters generate samples of the required distribution, proving beneficial in environments where the Gaussian assumption does not hold or multi-hypothesis (multi-modal) scenarios could arise. Nevertheless, complexity issues still surface as a large number of particles may be required to accurately represent the state-space distributions.

A BoW-based solution could alleviate some of the challenges faced by alternative schemes and gracefully address the issues listed in the above specification. The use of quantised SIFT features ensures robust matching under varying conditions and viewpoints, while a pre-defined vocabulary tree elegantly combats concerns on complexity and scalability. Cummins and Newman [9] find that a BoW-based SLAM system is capable of finding correct matches in the presence of considerable scene changes, permitting multiple pose hypotheses that can be appropriately filtered. Moreover, they were able to accurately detect loop closures in a database of images covering over 1,000 kilometres. The method is labeled as "appearance-only SLAM" as landmark positions are not estimated in metric coordinates, but are instead defined in an appearance space. This allows distinctive or unseen places to

be recognised and globally localised in the absence of any odometry information, providing a natural solution to common problems such as loop closure, multi-session mapping and the kidnapped robot, which tend to cripple metric SLAM solutions [9].

Chapter 2: Stitching and Reconstruction

In this chapter, a system is developed for offline reconstruction of a scene from photographs, building heavily on the research discussed in Section 1.3. As was illustrated in Figure 1.2, this module takes as input a corpus of images of a target scene. We divide the module into four sub-problems that are addressed in sequence, namely *indexing*, *stitching*, *reconstruction* and *bundle adjustment*. In Section 2.1 we introduce an image indexing system, which provides a preliminary step necessary for stitching and reconstruction. Indexing includes the extraction, quantisation and efficient storage of features from a database of images. The 2D stitching process (Section 2.2) then computes local image matches and their piecewise transformations by rapidly querying each database image against the index. This is used to generate a two-layered map: a *topological* link graph represents the connections between overlapping images, and a *metric* feature map plots the SIFT keypoint positions on the world plane. In Section 2.3, the 2D solution is extended to enable recovery of 3D camera poses by examining the scene geometry. Finally, errors in piecewise homography estimates and feature positions are jointly corrected using the bundle adjustment algorithm discussed in Section 2.4.

2.1 Image Indexing

A MATLAB-based program has been developed by Vedaldi[1] to index and query a collection of images. It closely follows the system described in [19], achieving near state-of-the-art performance, and has been used as a starting point for this project. The application is divided into two segments: the first is a one-off pre-computation stage (Section 2.1.1) to generate an index of images and visual words, summarised in Figure 2.1; the second querying stage (Section 2.1.2) takes an input image and returns a ranked list of matches from the database, as per Figure 2.3. In this Section, we detail the system architecture to understand how it interfaces with the stitching and reconstruction procedures developed in Sections 2.2 and 2.3.

2.1.1 Pre-Computation

Feature Extraction (Fig. 2.1a): For each image in the dataset, interest points are detected. The Scale Invariant Feature Transform [14] has been chosen as it is resilient to the effects of noise and is relatively unaffected by practical complications such as changes in scale or viewpoint, which weaken other methods. As it was originally designed for object recognition purposes, it is naturally suited to global localisation. It has been shown experimentally that maxima of the Laplacian-of-Gaussian give the best notion of scale, but this can be reasonably approximated by Difference-of-Gaussians.

[1] Available at: https://github.com/vedaldi/visualindex

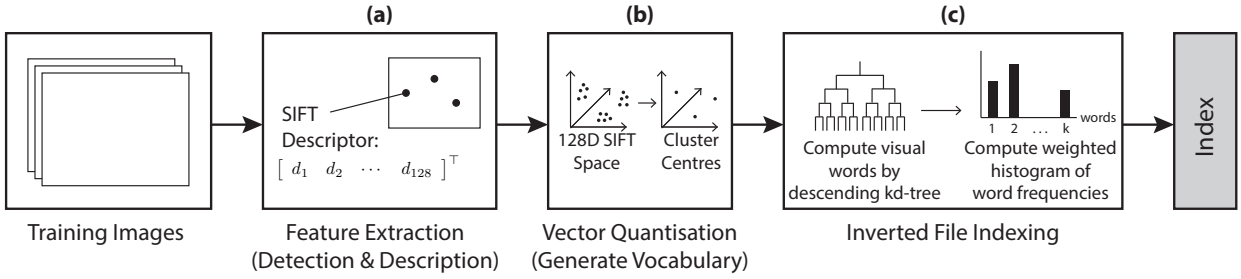


Figure 2.1: Pre-computation process

Scale invariance is thus realised by convolving the image with a 2D Gaussian kernel $G(x, y, \sigma)$ at different scales (σ) and computing the difference between the blurred images. The identified maxima and minima (called *scale-space extrema*) are consequently blob-like structures that are localised in both scale and space.

Rotational invariance is attained by assigning an orientation to each keypoint based on local image gradients. In the implementation, each feature's position, scale and orientation is encoded as an orientated ellipse. The SIFT "frame" or "keypoint" is thus specified by 6 floating point numbers:

$$\begin{bmatrix} x & y & a_{11} & a_{12} & a_{21} & a_{22} \end{bmatrix}^\top, \tag{2.1}$$

where (x, y) describes its position and a_{ij} are the elements of a 2×2 matrix that maps the unit circle to the ellipse. The local gradient information is then rotated to align with the frame's orientation and is used to generate the *descriptor*. The region about the keypoint is divided into a 4×4 grid, where each cell contains a histogram of gradients with 8 bins. This gives a descriptor vector with 128 gradient values:

$$\begin{bmatrix} d_1 & d_2 & \cdots & d_{128} \end{bmatrix}^\top \tag{2.2}$$

Similarity between SIFT descriptors is typically measured using the Euclidean distance. However, Arandjelović and Zisserman [3] note that replacing this with a square root (Hellinger) kernel yields substantial improvements across all stages of the application. The motivation behind this is to increase sensitivity of the similarity measure to smaller bin values, whereas the Euclidean distance between SIFT vectors tends to be dominated by large bin values. Conversion can be achieved via a straightforward transformation of descriptors from SIFT space to "RootSIFT" space: SIFT vectors are L_1-normalised and their element-wise square root is taken. This operation is implemented in a single line of code, after which the Euclidean distance computed between RootSIFT vectors is analogous to the Hellinger kernel for the original SIFT descriptors. Our baseline system makes use of this simple conversion throughout the pipeline.

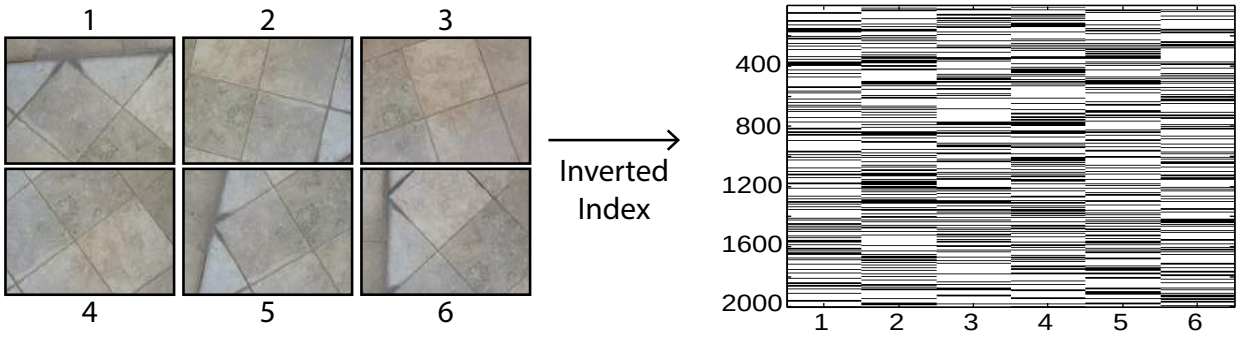


Figure 2.2: Visual words are extracted from images and indexed into an inverted file. The sparsity matrix of the inverted file in MATLAB is shown on the right (black cells indicate that an image contains a given word). In the above example, 6 images are indexed using a vocabulary of 2000 words. The words are weighted using the tf-idf scheme such that more contextually informative words take greater precedence during matching.

Vector Quantisation (Fig. 2.1b): Continuous SIFT space is then discretised by clustering a random sample of representative descriptor vectors using approximate k-means. This unsupervised training process builds the visual vocabulary in the form of a kd-tree, and the nearest-neighbour visual word to a given descriptor is obtained by descending the tree. For mapping purposes, the vocabulary size (number of cluster centres) chosen was typically between 10 and 20 times less than the number of features. This is typically the longest step of the pre-computation phase, but the dictionary (kd-tree) is defined once and for all.

Inverted File Indexing (Fig. 2.1c): All extracted descriptors are quantised by assigning to them the single-integer label of the nearest cluster centroid. This results in a highly compact image representation, where each image is characterised as a weighted histogram of visual word frequencies using the tf-idf scheme described in Section 1.3. For rapid querying the database is arranged as an inverted file [18], which maintains for each visual word the list of images in which the word occurred. In MATLAB, the inverted file exists in the form of a sparse $n \times m$ matrix for n words and m images, as shown in Figure 2.2.

2.1.2 Querying

BoW-based Image Matching (Fig. 2.3a): Once visual words have been computed for a query image, a tf-idf-weighted histogram $\mathbf{h}_{query}$ is generated in the form of an n-vector, where n is the vocabulary size. The "inverted file" matrix (described above) containing all word occurrences in each image is weighted to give the index histogram. This is a sparse $n \times m$ matrix $\mathbf{H}_{index} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_m \end{bmatrix}$, where $\mathbf{h}_i$ is the ith image histogram and m is the number of indexed images. The matching scores are then calculated by taking the dot product of the query histogram with each of the image histograms $\mathbf{h}_{query} \cdot \mathbf{h}_i$ (histograms are L_2-normalised so that their dot product is equal to their cosine similarity).

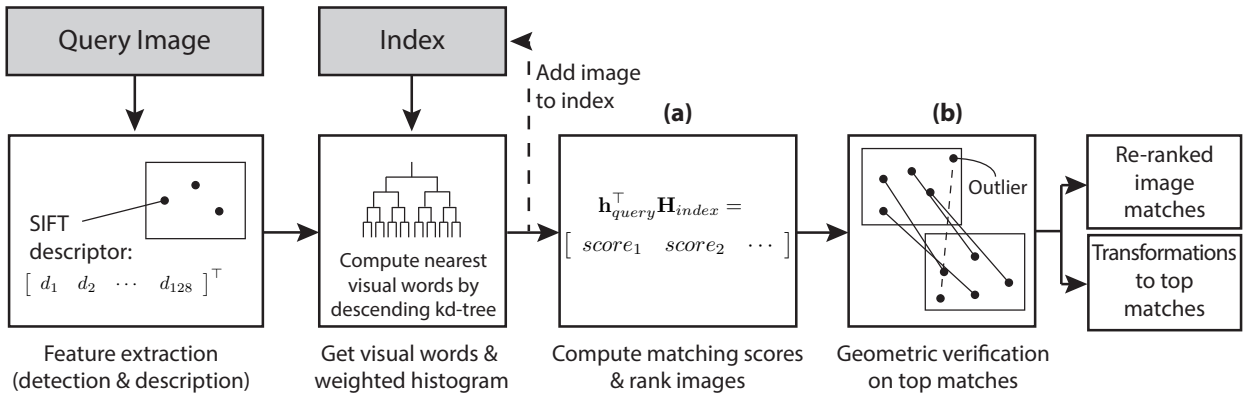


Figure 2.3: Querying process

This can be written as:

$$\mathbf{s} = \mathbf{h}_{query}^\top \mathbf{H}_{index} = \begin{bmatrix} score_1 & score_2 & \cdots & score_m \end{bmatrix} \tag{2.3}$$

Geometric Verification (Fig. 2.3b): A system that uses only the bag-of-words model is plausible, but for the purposes of large-scale mapping a geometric post-verification stage is fundamental to ensure that matched images satisfy epipolar constraints. Cummins and Newman [9] note that the impact of this process is particularly noticeable as index size increases: for a dataset covering 70km of physical space it is helpful, but for a 1000km dataset it is essential. In textured environments images are likely to share many common features due to repetitive structures, making BoW matching alone prone to false positives.

Instead, BoW-based matching scores are sorted and the top r images with scores $\mathbf{s}_r$ are tested for spatial consistency; r is the user-defined "re-rank depth" and is typically set between 5 and 50 depending on index size. For rapid verification, some systems assume the transformation between poses is a pure rotation (described by 4 parameters) [9]. However, as we wish to recover accurate poses for mapping and localisation, we relate keypoints using an affine (6-parameter) homography matrix $\mathbf{H}$. The RANSAC (Random Sampling Consensus) algorithm randomly selects candidate interest point matches from the BoW assignment. For each match, $\mathbf{H}$ can be directly estimated as the transformation between the two corresponding SIFT frames:

$$\mathbf{H} = \mathbf{F}_{model}\mathbf{F}_{query}^{-1}, \text{ where } \mathbf{F} = \begin{bmatrix} a_{11} & a_{12} & x \\ a_{21} & a_{22} & y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

a_{ij} and (x, y) form an individual frame as defined in (2.1). Corresponding points in the query and model images $\mathbf{x}_{query} \leftrightarrow \mathbf{x}_{model}$ are ideally related by $\mathbf{x}_{model} = \mathbf{H}\mathbf{x}_{query}$ in homogeneous co-ordinates,

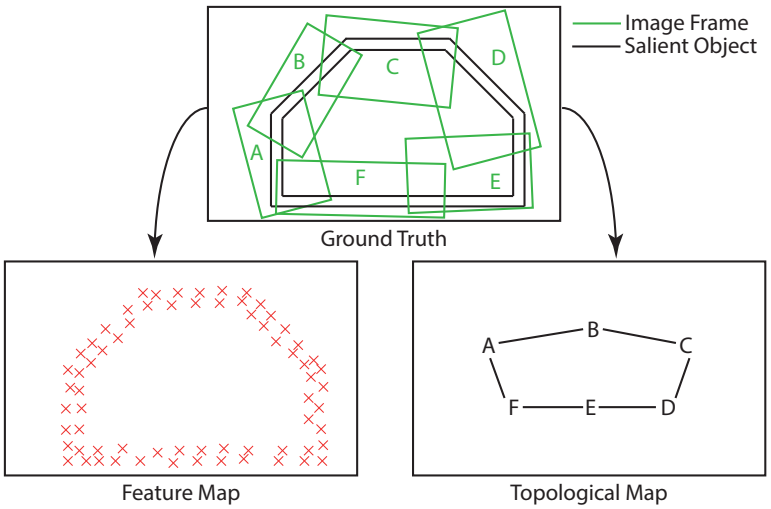


FIGURE 2.4: The world is represented on one level by the topological relationship between images, and on another level by the metric distribution of features. This yields an environment model with two distinct and complementary levels of abstraction.

and as such each candidate estimate of $\mathbf{H}$ can be tested by applying this relation and counting the number of matched points that fall within a given tolerance region ($\|\mathbf{x}_{model} - \mathbf{H}\mathbf{x}_{query}\| < tol$). For the value of $\mathbf{H}$ with the largest support, $\mathbf{H}$ is re-computed using all associated inliers.

The maximum number of inliers for each image tested is stored in an r-vector **inliers**, and the matching scores are updated using the assignment:

$$\mathbf{s}_r \leftarrow \mathbf{s}_r + \mathbf{inliers} \tag{2.5}$$

Finally, the scores are re-ranked to give the top matches. The rationale behind this approach is to apply a coarse but computationally lightweight search algorithm (BoW) to the entire index, followed by an accurate but complex verification stage on a small subset of images. For localisation and mapping we are concerned with the accuracy of $\mathbf{H}$, and as such it may be appropriate to perform spatial verification on every single feature correspondence (rather than a random subset) to maximise the probability of correct recovery.

2.2 Image Matching Problem: Stitching in 2D

In the previous section we introduced a bag-of-words-based indexing system that enables us to efficiently determine pairs of overlapping images. The following section builds on this capability by constructing a global map of the environment and stitching together the imaged scene (assumed to be planar), such that all images are aligned to a common reference frame. Metric, topological and hybrid mapping approaches have been adopted in the past to represent the environment. Metric maps permit pose recovery in the form (x, y, α) in two dimensions or $(x, y, z, \alpha, \beta, \gamma)$ in three dimensions, where (α, β, γ)

represent pitch, roll and yaw. Conversely, in topological maps position is defined only in relation to high-level "locations", often linked by a correspondence graph. Whilst metric approaches, which are typically feature-based, offer greater precision, they are also more vulnerable to drift and data association problems that make long-term global consistency difficult to achieve. Maintaining the topology of an environment – that is, understanding how different places and landmarks are related – better handles the problem of global coherence; however, it also offers less precision due to significant discretisation of the localisation space [27].

Integration of the topological and metric paradigms could help to compensate for the weaknesses of each individual approach [27]. A environment model characterised by two levels of abstraction, as illustrated in Figure 2.4, is therefore proposed. The relationships between individual images in the database are compactly represented by a **link graph**: each image occupies a node and edges signify that two images physically overlap. Each edge is associated with an estimated homography matrix relating the connected vertices. The **feature map** plots all extracted SIFT keypoints within a single 2D global co-ordinate system. Associated local features are merged into a single representative keypoint in the map.

This dual representation decouples many of the problems faced in mapping: the correspondence graph straightforwardly handles issues such as loop closure and provides a simple solution to path planning in the form of a shortest distance graph search. The feature map is initially constructed using the graph, but subsequently enables finer adjustment of image and feature relations such that local geometric information is globally consistent (see Section 2.4); these fine-tunings are fed back into the graph. Features also offer a better understanding of the physical composition of the scene, and can provide efficiency improvements such as the map segmentation approach developed in Chapter 3.

2.2.1 Topological Mapping (Link Graph)

The undirected graph is constructed using the process demonstrated in Figure 2.5. In MATLAB, it is represented by an $n \times n$ symmetric adjacency matrix $\mathbf{G}$ for n indexed images; if $g_{ij} = 1$ then an edge exists between nodes (images) with IDs i and j. $\mathbf{G}$ is therefore highly sparse for a large index. In the implementation, the adjacency matrix, image-to-image feature matches and estimated transformations are stored in a structure array "*cor*", which is computed as shown in Listing 2.1.

The code runs through each model image entry, querying it against the index to determine which images are spatially linked to it. After geometric verification, a judgment must be made based on matching scores as to which images physically overlap with the query image, such that they both capture part of the same planar body. It was found experimentally that setting an absolute threshold alone is not robust: the number of features matches tends to vary with texture and image resolution,

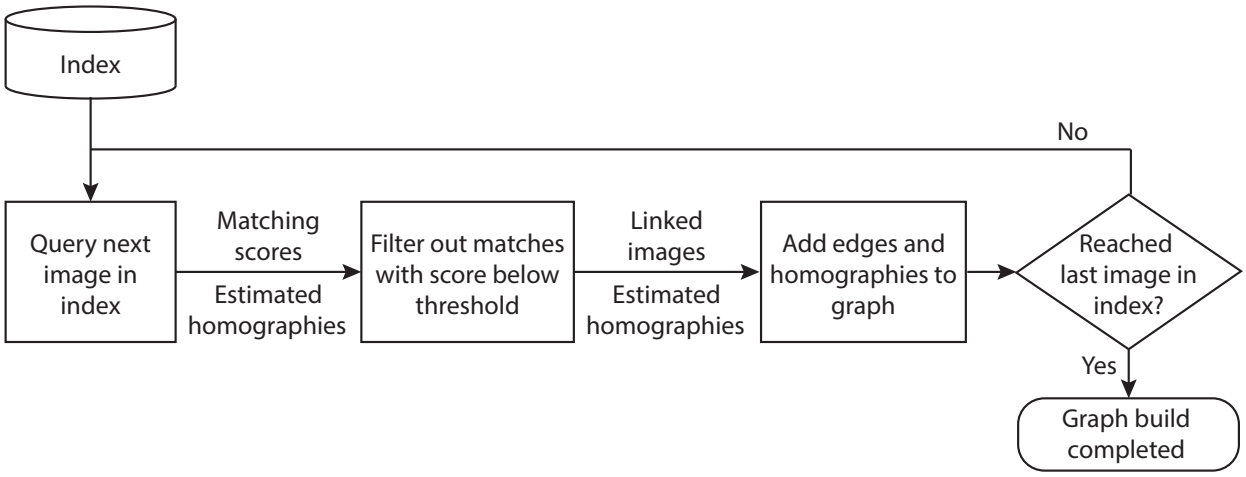


Figure 2.5: Process map showing how the link graph is constructed

which in turn impacts the number of inliers. However, assigning an additional threshold equal to some percentage of the best (non-self) matching score provided better results, as it introduced context-specific information. Thus, the condition for the kth ranked score s_k to be judged as overlapping is:

$$s_k > t_{abs} \text{ and } s_k > t_{perc} s_2 \tag{2.6}$$

Note that s_1 is the matching score for the query image against itself (since it has already been indexed), and is therefore discarded. Values of $t_{abs} = 30$ and $t_{perc} = 0.4$ proved successful for a wide range of textures (wood, marble, concrete) and image resolutions (from 1000px width to 5000px width).

Figure 2.6 demonstrates the outcome of the algorithm given 12 model images. Notice that the link graph naturally permits multi-session mapping: the training images depict two distinct places, causing the nodes to cluster into scene-specific cliques. Queries are computed over the entire index, allowing localisation and mapping across multiple different maps.

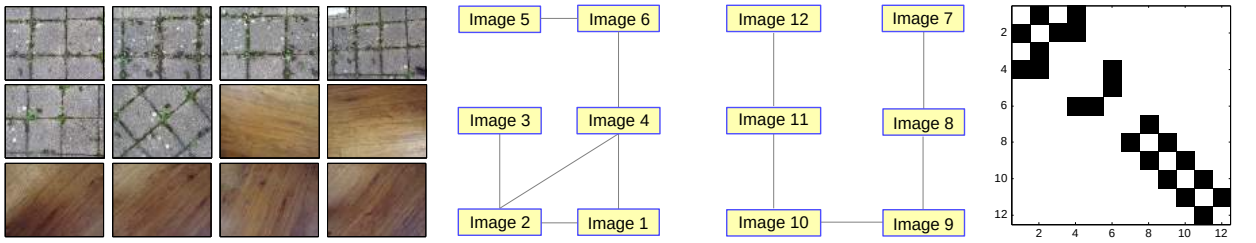
Once the graph is constructed, all images must be positioned within a common co-ordinate frame in preparation for stitching. This is realised by arbitrarily selecting a reference image to coincide with the global frame – a popular technique in literature [24, 23]. Pixels in the reference image are transformed to world co-ordinates by the identity matrix. For all other images, global transformations are obtained by traversing the graph along a minimum spanning tree (MST) and cascading all pairwise homographies that branch from the reference node. Let us denote the estimated homography from image i to j as $\mathbf{H}_i^j$, such that in homogeneous co-ordinates pixels are related by $\mathbf{x}_j = \mathbf{H}_i^j \mathbf{x}_i$. Further, let the MST from the reference node be represented by an ordered set $O = \{o_1, o_2, \ldots, o_n\}$, where o_k is the ID of the kth image in the tree, o_1 is the ID of the reference image, and n is the number of vertices connected (indirectly) to the reference image. For an arbitrary image with ID o_k, the homography to the reference frame can be calculated as:

```
num_images = length(model.index.ids);
init_cell = cell(1, num_images);
cor = struct('id', model.index.ids, 'img_matches', init_cell, 'scores', init_cell, ...
    'feature_matches', init_cell, 'H', init_cell, 'adjacency', eye(num_images));

for i = model.index.ids % Iterate through indexed image IDs
    [match_ids, scores, result] = visualindex_query(model, i); % Query image i
    score_thresh = scores(2) * opts.percentThresh; % Set threshold value
    % Note that scores(1) is the score for the query image against itself
    goodmatch = find(scores > score_thresh & scores > opts.numThresh ...
        & ids ~= result.query);
    cor.img_matches{i} = match_ids(goodmatch);
    cor.scores{i} = scores(goodmatch);
    cor.H{i} = result.H(goodmatch);
    cor.feature_matches{i} = result.matches(goodmatch);
    cor.adjacency(i, ids(goodmatch)) = 1;
end

% Find edges that are not bi-directional
[im_ids, match_ids] = find(cor.adjacency .* cor.adjacency' ~= cor.adjacency);
% Remove them from adjacency matrix and cor struct
```

Listing 2.1: MATLAB code to build the correspondence structure *cor* by iteratively querying each image in the index. *model* is a structure array containing the indexed images, their histograms, and the kd-tree. *visualindex_query* is a function provided by the indexing package, which queries an image and returns: (1) the IDs of matched images; (2) their corresponding matching scores; and (3) a structure *result* containing the estimated transformations **H** from geometric verification and the specific feature matches between the query and model images.



(a) *Indexed images* **(b)** *Link graph (built using biograph function)* **(c)** *Adjacency matrix (white fields are zero)*

Figure 2.6: Link graph and adjacency matrix generated from a set of 12 model images.

$$\mathbf{H}_{o_k}^{o_1} = \mathbf{H}_{o_2}^{o_1}\mathbf{H}_{o_3}^{o_2}\dots\mathbf{H}_{o_k}^{o_{k+1}} \tag{2.7}$$

A complication introduced by combining piecewise homographies is that drift errors accumulate. Consequently, a breadth-first approach is used when computing the MST to minimise this effect. Edges can also be weighted by the average matching scores between connected images. In MATLAB, the *graphtraverse* function is utilised to calculate the MST. Thereafter, a mosaic is simply produced by applying the global transformation to each image, yielding the outcome in Figure 2.7.

The central projection of points from the world plane to the image plane is described by a perspective transformation which, by definition, does not preserve parallel lines. This results in perspective distortion if the camera is not fronto-parallel to the world plane. The degree of perspective distortion in the mosaic is therefore dependent on the choice of reference image: distortion in the reference image will propagate throughout the stitching, resulting in errors that exaggerate further away from the origin. Although these errors are reversed by optimisation algorithms such as those described in Section 2.4, it is desirable to select a reference image that is approximately fronto-parallel to the world plane and centrally located within the graph (such that the MST depth is minimised). Extension to 3D recovery also overcomes the perspective problem, as described in Section 2.3.3.

2.2.2 Metric Mapping (Feature Map)

Images from a single camera cannot measure the scale of the world – there is nothing to distinguish a small nearby object from a large distant one. This inherent scale-depth ambiguity cannot be overcome without additional information about the environment, such as the physical size of an object in view [10]. Nevertheless, it is possible to construct a feature map up to an unknown scale factor. In our case, scale is dictated by the pixel dimensions of the reference image.

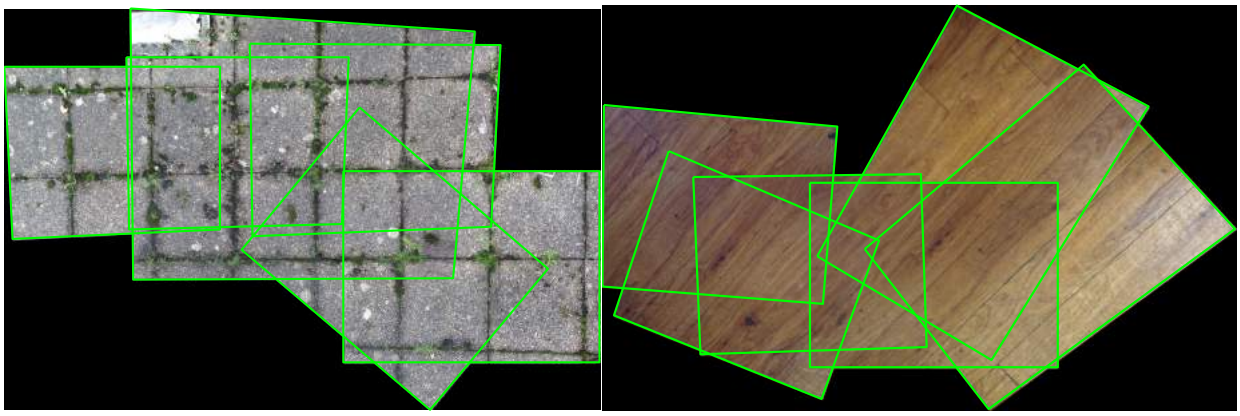


Figure 2.7: Stitching results from the dataset in Figure 2.6a. A separate mosaic is produced for each isolated cluster of nodes.

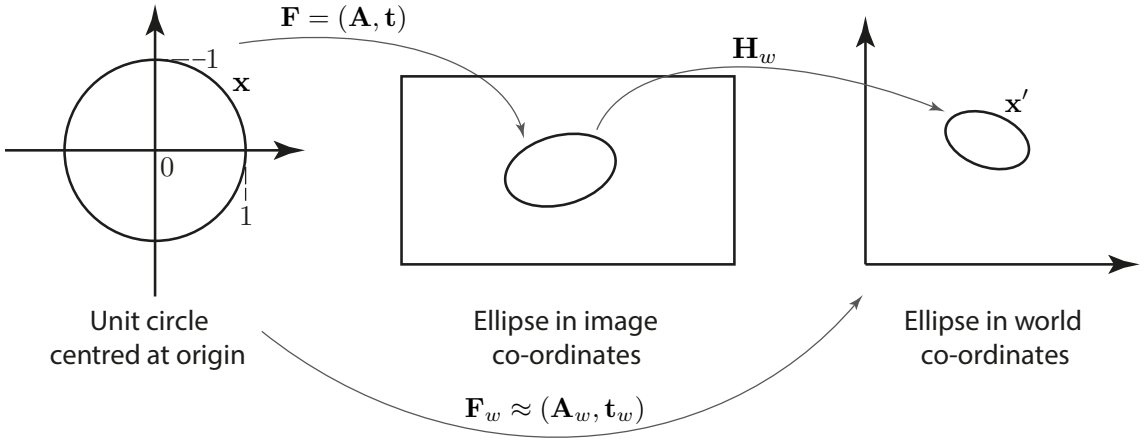


FIGURE 2.8: The unit circle must undergo multiple affine transformations to form the oriented elliptical SIFT frame in world co-ordinates.

Following from the calculation of each image's global position, constructing the feature map is a straightforward procedure of transforming local SIFT frames to world co-ordinates. The details of this transformation are discussed next, followed by an outline of how they are stored in our MATLAB implementation.

Transforming SIFT Keypoints: As described in Section 2.1.1, SIFT frames are orientated ellipses characterised by six parameters that form an affine transformation:

$$\mathbf{F} = \left[\begin{array}{cc} \mathbf{A} & \mathbf{t} \\ \hline \mathbf{0}^\top & 1 \end{array}\right], \tag{2.8}$$

where $\mathbf{A}$ defines the mapping from the unit circle to the orientated ellipse centred at the origin, and $\mathbf{t}$ is a translation vector corresponding to the centre of the ellipse. Given the homography $\mathbf{H}_w$ from an image to the world, we are required to transform the local keypoint $\mathbf{F} \rightarrow \mathbf{F}_w$ accordingly. From Figure 2.8 it can be seen that:

$$\mathbf{F}_w = \mathbf{H}_w \mathbf{F} \tag{2.9}$$

However, the frame resulting from this calculation may not be affine (the lower left-hand terms of $\mathbf{F}_w$ are non-zero) as $\mathbf{H}_w$ is not necessarily affine. Furthermore, the transformation $\mathbf{F}_w$ from the unit circle to the ellipse in world co-ordinates is not unique.

To enforce affinity of $\mathbf{F}_w$ we can instead calculate a transformation $(\mathbf{A}_w, \mathbf{t}_w)$ that approximates to $\mathbf{F}_w$. The mapping $\mathbf{x} \rightarrow \mathbf{x}'$ from the unit circle to the oriented ellipse in world co-ordinates (as per Figure 2.8) can be described in homogeneous co-ordinates as $\begin{bmatrix} \mathbf{x}' \\ 1 \end{bmatrix} = \mathbf{F}_w \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$. In inhomogeneous co-ordinates it is denoted by a vector function $\mathbf{x}' = \mathbf{G}(\mathbf{F}_w, \mathbf{x})$, where:

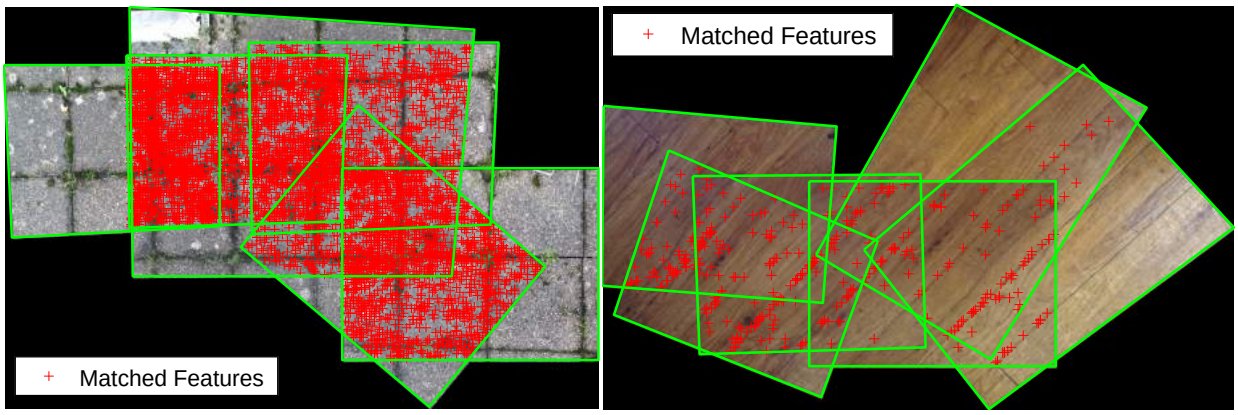


FIGURE 2.9: Feature maps for the mosaics in Figure 2.7. Notice the difference in the number of matched features, partly due to the fact that the left images have more distinct visual characteristics, thus justifying the filtering scheme in (2.6). The lack of matched features on the right signals the possibility of having false positive image matches and poor homography estimates as the index grows.

$$\mathbf{G}\left(\mathbf{F}_w, \mathbf{x}\right) = \frac{\mathbf{F}_{11}\mathbf{x} + \mathbf{f}_{12}}{\mathbf{f}_{21}^\top \mathbf{x} + f_{22}}, \qquad \mathbf{F}_w = \left[\begin{array}{cc} \mathbf{F}_{11} & \mathbf{f}_{12} \\ \hline \mathbf{f}_{21}^\top & f_{22} \end{array}\right] \tag{2.10}$$

The objective is to approximate this operation to a single affine transformation:

$$\forall\, \mathbf{x} \in \mathbb{R}^2 : \mathbf{x}' = \mathbf{G}\left(\mathbf{H}_w\mathbf{F}, \mathbf{x}\right) \approx \mathbf{A}_w\mathbf{x} + \mathbf{t}_w, \tag{2.11}$$

Combining (2.11) and (2.10) we can obtain a good approximation for the parameters $(\mathbf{A}_w, \mathbf{t}_w)$ by performing a first order Taylor expansion of $\mathbf{x}'$ about $\mathbf{x} = \mathbf{0}$:

$$\mathbf{A}_w\mathbf{x} + \mathbf{t}_w = \mathbf{G}\left(\mathbf{F}_w, \mathbf{0}\right) + \left.\frac{\partial \mathbf{G}\left(\mathbf{F}_w, \mathbf{x}\right)}{\partial \mathbf{x}^\top}\right|_{\mathbf{x}=\mathbf{0}} \mathbf{x} \tag{2.12}$$

By inspection, we observe that:

$$\mathbf{t}_w = \mathbf{G}\left(\mathbf{F}_w, \mathbf{0}\right) \text{ and } \mathbf{A}_w = \left.\frac{\partial \mathbf{x}'}{\partial \mathbf{x}^\top}\right|_{\mathbf{x}=\mathbf{t}} \tag{2.13}$$

The positions of global feature frames are unchanged by this approximation, but scale and orientation are adjusted to adhere to the affinity constraint. This adjustment is important, as global SIFT frames will later be used to localise query images within the map (see Chapter 3). Applying the approximate transformation to all local features gives the result in Figure 2.9. Note that although only matched features are displayed, unmatched frames are also computed.

Adding Features To The Map: In the implementation, a structure array "*world*" collects relevant information for the feature map. Three important variables are generated. First, *world.features_global*

is an $8 \times m$ array, where m is the number of unique global features and each column takes the form:

$$\left[\begin{array}{cccccccc} \text{global\_feat\_id} & \text{num\_matched\_feats} & x & y & a_{11} & a_{12} & a_{21} & a_{22} \end{array} \right]^\top \tag{2.14}$$

The latter six elements, which constitute a SIFT frame, are deduced by iteratively transforming all local keypoints matched to the global feature and taking their element-wise mean. This provides a reasonable guess for the feature's position prior to optimisation (discussed in Section 2.4). If a local feature is found to have no global matches, a new global ID is initialised to which it is assigned.

Second, *world.feature_ map* is a $3 \times n$ array, where n is the total number of local features across all images and $n \geqslant m$. Each column is structured as $\left[\begin{array}{ccc} \text{local\_feat\_id} & \text{img\_id} & \text{global\_feat\_id} \end{array} \right]^\top$, and serves to map each local feature to a global feature. However, this representation does not permit efficient retrieval of all local features allocated to a given global ID – functionality which is essential for the optimisation algorithms developed later. Since searching an unsorted array has complexity $O(n)$, applying linear search to *world.feature_ map* becomes sluggish for large maps. Consequently, an inverted index is constructed as an m-column sparse matrix, such that column k contains the local IDs of all features mapped to global feature k; this is stored in the variable *world.feature_ index*. Hence, the map is built incrementally by running through each image in the minimum spanning tree and transforming and allocating its local SIFT keypoints.

2.3 Geometric Problem: Reconstruction in 3D

Thus far we have stitched the imaged scene in two dimensions by arbitrarily selecting a reference image and using information about pairwise homographies to relate all images to the "world". However, there is no guarantee that the reference image is fronto-parallel to the ground plane, meaning that undesirable perspective distortions can arise in the mosaic. In this section we examine the camera and scene geometry, exploiting the planar assumption to represent the 2D stitching in 3D space. By simply computing the transformation from the reference image to the ground plane (as per Section 2.3.3) the perspective effect can be resolved and camera poses can be recovered.

2.3.1 Camera Geometry

We assume the standard pinhole camera model, which considers the central projection of a 3D point in the camera co-ordinate system $\mathcal{C}$ onto the image plane. This is described by a linear transformation $\mathbf{P} = \mathbf{K}[\mathbf{I}|\mathbf{0}]$ from the projective space $\mathbb{P}^3$ to the projective plane $\mathbb{P}^2$, where $\mathbf{K}$ is a 3×3 matrix representing the camera's internal calibration parameters. Furthermore, if the scene is defined within a world co-ordinate frame $\mathcal{W}$, then an additional extrinsic roto-translation matrix $[\mathbf{R}|\mathbf{t}]$ is required to align the

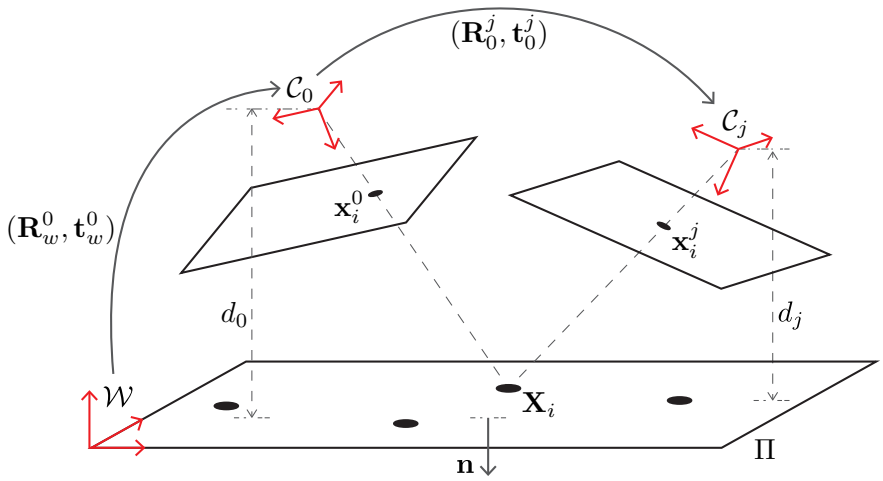


Figure 2.10: Two views of the planar structure. $\mathbf{X}_i$ is the position of the global feature in inhomogeneous metric world co-ordinates, whereas $\mathbf{x}_i^0, \mathbf{x}_i^j$ are the projections of $\mathbf{X}_i$ onto the image planes, which are defined in pixel co-ordinates. The two camera poses $\mathcal{C}_0, \mathcal{C}_j$ are related by a Euclidean homography $\mathbf{G}_0^j$ as described in (2.18).

world and camera frames. Thus, the mapping from a point in the world $\mathbf{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^\top$ to the image plane $\mathbf{x} = \begin{bmatrix} x & y & 1 \end{bmatrix}^\top$ is described using homogeneous co-ordinates by the 3×4 projection matrix $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ [13]:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \propto \underbrace{\begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} \mathbf{R}_w^{3\times 3} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.15}$$

where $\propto$ denotes equality up to an unknown scale factor, (f_x, f_y) is the focal length of the camera in pixel units, s is the skew factor accounting for non-rectangular pixels, and (x_0, y_0) is the principal point offset. $\mathbf{K}$ is thus a non-singular affine transformation from the metric image plane to pixel co-ordinates.

2.3.2 Scene Geometry

Until now only local information extracted from images has been utilised to generate a 2D map. Attaining full 3D reconstruction requires information about the world and camera geometry to be incorporated into the environment model. To understand this geometry, consider a planar scene observed from multiple views (illustrated in Figure 2.10), whereby the camera frames are related by rigid transformations (roto-translations) in 3D Euclidean space. In general, the co-ordinate transformation from camera frame 0 (the reference) to j can be described by [15]:

$$\mathbf{X}_i^j = \mathbf{R}_0^j \mathbf{X}_i^0 + \mathbf{t}_0^j, \tag{2.16}$$

where $\mathbf{X}_i^0, \mathbf{X}_i^j \in \mathbb{R}^3$ are the co-ordinates of world feature $\mathbf{X}_i$ relative to $\mathcal{C}_0$ and $\mathcal{C}_j$ respectively. Given that the scene is planar, we can specialise this relation. Let $\mathbf{n}_0$ denote the unit normal to the world plane in camera 0's reference frame. Noting that the projection of $\mathbf{X}_i^0$ onto vector $\mathbf{n}_0$ gives the distance d_0 from the world plane to the optical center of camera 0, we write:

$$\mathbf{X}_i^0 \cdot \mathbf{n}_0 = d_0 \;\Rightarrow\; \frac{\mathbf{n}_0^\top \mathbf{X}_i^0}{d_0} = 1, \quad \forall\, \mathbf{X}_i \in \Pi \tag{2.17}$$

Substituting (2.17) into (2.16) gives the relation:

$$\mathbf{X}_i^j = \mathbf{R}_0^j \mathbf{X}_i^0 + \mathbf{t}_0^j \frac{\mathbf{n}_0^\top \mathbf{X}_i^0}{d_0} = \mathbf{G}_0^j \mathbf{X}_i^0 \tag{2.18}$$

To summarise: (2.16) shows that for any 3D point $\mathbf{X}_i$ in a scene, the change in co-ordinates between camera frames is a rigid-body transformation. Our assumption that the scene is planar imposes the additional constraint in (2.17) that all features share the same depth relative to camera 0. The transformation therefore reduces to a Euclidean homography matrix $\mathbf{G}_0^j = \mathbf{R}_0^j + \frac{\mathbf{t}_0^j \mathbf{n}_0^\top}{d_0}$. Having computed the *projective* homography $\mathbf{H}_0^j$ from feature matches, we can recover $\mathbf{G}_0^j$ up to an unknown scale factor γ using the relation $\mathbf{H} = \gamma \mathbf{K}\mathbf{G}\mathbf{K}^{-1}$. As the second eigenvalue of $\mathbf{G}_0^j$ is equal to unity, we can determine the normalising factor as $\gamma = \text{median}\left(\text{singular values}\left(\mathbf{K}^{-1}\mathbf{H}_0^j\mathbf{K}\right)\right)$ assuming the intrinsic camera parameters are known [16].

The representation in (2.18) has a number of important implications. First, $\mathbf{G}_0^j$ depends both on motion parameters $(\mathbf{R}, \mathbf{t})$ as well as structure parameters $(\mathbf{n}, d)$ of the world plane. Second, due to scale-depth ambiguity in the $\mathbf{t}/d$ term we can only expect to recover the ratio of camera translation $\mathbf{t}$ to depth d. Some vision applications attain pose recovery by extracting the structure and motion variables from $\mathbf{G}_0^j$ – a process known as homography decomposition [16, 15]. This typically relies on singular value decomposition, although Malis and Vargas [16] provide analytical expressions for the poses. The approach is especially useful for applications such as [30] requiring multi-plane registration, where several planes must be localised and their normals estimated. However in the most general case it generates four distinct solutions, which can be reduced to two physically feasible alternatives by imposing the visibility constraint (all features must lie within the camera field of view). Further ambiguity is difficult to resolve without prior knowledge, especially when there is only one plane in view.

An Alternative Approach: A simpler formulation more compatible with single-plane structures is obtained by choosing the world co-ordinate system $\mathcal{W}$ to align with the ground plane, such that the x- and y-axes coincide with Π and the z-axis is perpendicular to it [24]. Consequently and without loss

of generality, this restricts all 3D feature points $\mathbf{X} = \begin{bmatrix} X & Y & 0 & 1 \end{bmatrix}^\top$ to lie along the plane $z = 0$ in the world frame. This simplification reduces the projection matrix in (2.15) to a planar homography. For camera j we write:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda \mathbf{K} \left[\begin{array}{c|c} \mathbf{R}_w^j & \mathbf{t}_w^j \end{array} \right] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \lambda \mathbf{K} \left[\begin{array}{ccc|c} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}_w^j \end{array} \right] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{2.19}$$

$$= \lambda \mathbf{K} \left[\begin{array}{cc|c} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}_w^j \end{array} \right] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{H}_w^j \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.20}$$

Homography matrix $\mathbf{H}_w^j \in \mathbb{R}^{3\times 3}$ projects points on the world plane to the image plane of camera j up to an unknown scale factor λ. The extrinsic parameters of camera j can be extracted from $\mathbf{H}_w^j$ assuming $\mathbf{K}$ is known *a priori*. The decomposition method described by Simon and Berger [23] exploits the orthonormality of the columns of rotation matrix $\mathbf{R}_w^0$: although the third column of $\mathbf{R}_w^0$ is omitted in (2.20), we recover it by applying the constraints of unit norm ($\|\mathbf{r}_1\| = \|\mathbf{r}_2\| = 1$) and orthogonality ($\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$). The rotation is therefore approximated by:

$$\mathbf{r}_1 = \lambda \mathbf{K}^{-1}\mathbf{h}_1, \qquad \mathbf{r}_2 = \lambda \mathbf{K}^{-1}\mathbf{h}_2, \qquad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{2.21}$$

where $\mathbf{H}_w^j = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}$ and $\lambda = \frac{1}{\|\mathbf{K}^{-1}\mathbf{h}_1\|} = \frac{1}{\|\mathbf{K}^{-1}\mathbf{h}_2\|}$. However, due to noise in the estimated homography the computed matrix $\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix}$ may not satisfy the orthonormality conditions of a rotation matrix. To impose the correct form, a new optimal rotation matrix $\mathbf{R}'$ is computed using singular value decomposition as described by [31]:

$$\mathbf{R} = \mathbf{U}\mathbf{S}\mathbf{V}^\top \Rightarrow \mathbf{R}' = \mathbf{U}\mathbf{V}^\top, \tag{2.22}$$

where $\mathbf{S} = diag(\sigma_1, \sigma_2, \sigma_3)$. This method solves for the rotation matrix $\mathbf{R}'$ that minimises the Frobenius norm of the difference $\mathbf{R}' - \mathbf{R}$. Finally, the normalising factor applied to the translation is taken as the mean of the two norms $\left\|\mathbf{K}^{-1}\mathbf{h}_1\right\|$ and $\left\|\mathbf{K}^{-1}\mathbf{h}_2\right\|$, given that they are not necessarily equal in practice:

$$\mathbf{t}_w^j = \frac{2\mathbf{K}^{-1}\mathbf{h}_3}{\|\mathbf{K}^{-1}\mathbf{h}_1\| + \|\mathbf{K}^{-1}\mathbf{h}_2\|} \tag{2.23}$$

2.3.3 From 2D to 3D: Computing $\mathbf{H}_w^0$ and Correcting Perspective

The previous section outlined an approach to recovering the extrinsic camera parameters for a given image j. However, this involves decomposing a homography $\mathbf{H}_w^j$ between the image and world planes that has not yet been computed. In order to efficiently obtain $\mathbf{H}_w^j \; \forall \; j$ we can exploit the topological network devised in Section 2.2.1. In (2.7) it was shown that for any arbitrary image in the map the transformation to the reference image can be calculated by cascading pairwise homographies. In order to extend this representation from 2D to 3D we require only one additional homography, $\mathbf{H}_w^0$, from the world plane to the reference image plane. Thereafter, having computed the camera displacement for a single reference image, we can compute the extrinsic parameters of all other frames by applying the following relation:

$$\mathbf{H}_w^j = \mathbf{H}_0^j \mathbf{H}_w^0 \tag{2.24}$$

Thus, the problem of directly measuring the transformation to the world plane need only be tackled for a single reference image. This is achieved offline using the Direct Linear Transform and involves manually selecting four points in the image that correspond to rectilinear co-ordinates in $\mathcal{W}$ [13, 24]. Although we do not know the true world co-ordinates of the points $\begin{bmatrix} X & Y & 1 \end{bmatrix}^\top$ in (2.20), we can parameterise the unknown shape S as follows, assuming that it is a rectangular structure:

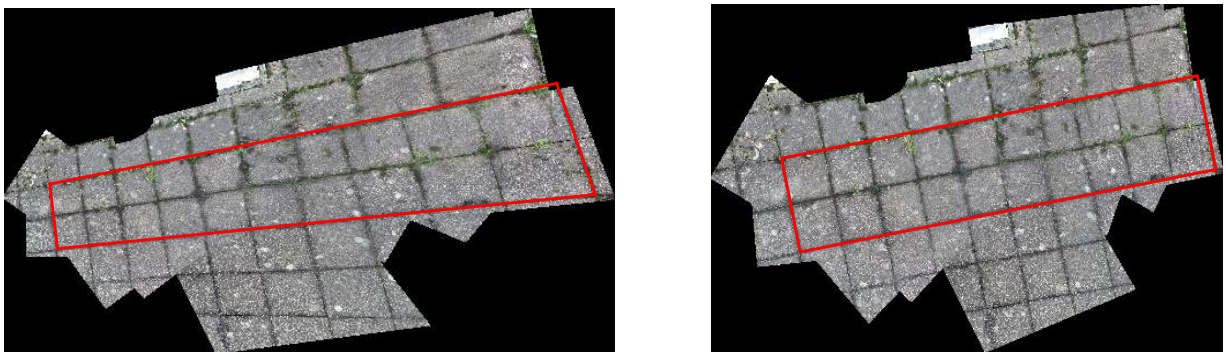
$$\mathbf{S} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & \alpha & \alpha \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.25}$$

Each column of $\mathbf{S}$ corresponds to a homogeneous 2D point on the world plane, where α is the unknown aspect ratio of the rectangular shape. This can then be divided into a scaling component $\mathbf{S}_\alpha$ and structural component $\mathbf{S}_s$:

$$\mathbf{S} = \mathbf{S}_\alpha \mathbf{S}_s = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.26}$$

The four points selected in the reference image can be mapped to the co-ordinates $(0,0)$, $(1,0)$, $(1,\alpha)$, $(0,\alpha)$ in the world plane. Let us assume initially that $\alpha = 1$. For the four corresponding pairs of inhomogeneous points $(X,Y) \leftrightarrow (x,y)$ measured from the world and image respectively, we can write:

$$x\left(h_{31}X + h_{32}Y + h_{33}\right) \;\; = \;\; h_{11}X + h_{12}Y + h_{13}$$



(a) *Region selected*

(b) *Region transformed to rectilinear co-ordinates*

FIGURE 2.11: Fixing perspective distortion in a mosaic of 25 images. Two pairs of parallel lines are identified when the user selects points that should lie on a rectangle (left). These are used to compute the rectified image (right).

$$y\left(h_{31}X + h_{32}Y + h_{33}\right) \quad = \quad h_{21}X + h_{22}Y + h_{23}$$

Equivalently, we can write the matrix equation:

$$\underbrace{\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1 \\ & & & & & \vdots & & & \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -x_4Y_4 & -x_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -y_4X_4 & -y_4Y_4 & -y_4 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{32} \\ h_{33} \end{bmatrix}}_{\mathbf{h}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \tag{2.27}$$

where $\mathbf{h}$ is the vertical stacking of the elements of $\mathbf{H}$, which maps the unit square to the set of selected points (x, y). Points must be chosen such that $\mathbf{A}$ has full rank, meaning that no 3 points can be collinear. The solution $\mathbf{h}$ is then simply the null space of $\mathbf{A}$ and can be computed using singular value decomposition [13].

2.3.3.1 Non-Unit Aspect Ratio

The structure $\mathbf{S}$ is unlikely to be a perfect square in practice. It can be seen from (2.26) that the effect of a non-unit aspect ratio is to pre-multiply the homogeneous 2D world co-ordinates by $\mathbf{S}_\alpha = \text{diag}(1, \alpha, 1)$. Hence the solution $\mathbf{H}$ from (2.27) can be written as [24]:

$$\mathbf{H} = \mathbf{H}_w^0 \mathbf{S}_\alpha = \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

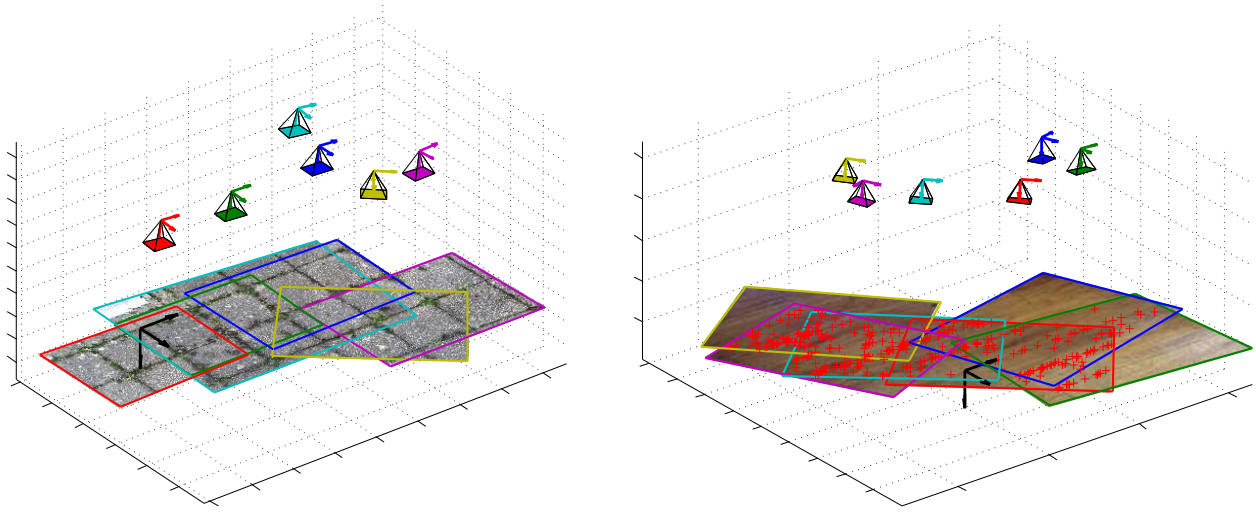


Figure 2.12: Camera poses recovered for the dataset in Figure (2.6a). The right plot also shows the estimated global positions of matched features in red.

$$\therefore \mathbf{K}^{-1}\mathbf{H} = \begin{bmatrix} \mathbf{r}_1 & \alpha\mathbf{r}_2 & \mathbf{t} \end{bmatrix} \tag{2.29}$$

We can therefore calculate the aspect ratio when $\mathbf{K}$ is known as $\alpha = \frac{\|\mathbf{h}_2\|}{\|\mathbf{h}_1\|}$, where $\mathbf{h}_1, \mathbf{h}_2$ are the first and second column of $\mathbf{K}^{-1}\mathbf{H}$ respectively. Finally, we can compute $\mathbf{H}_w^0 = \mathbf{H}\mathbf{S}_\alpha^{-1}$. Applying the inverse of $\mathbf{H}_w^0$ to the entire image also yields the rectified mosaic shown in Figure 2.11b. Figure 2.12 demonstrates the outcome of 3D pose recovery in the MATLAB implementation. The original structure array *cor*, described in Section 2.2.1, is updated to include the transformations of each image to the world's co-ordinate frame.

2.4 Bundle Adjustment

Thus far, global registration has involved cascading pairwise transformations along a minimum spanning tree from some reference image, and then relating the reference and world planes. However, drift errors can quickly accumulate as there is no backward correction of parameter and feature estimates when new information is obtained. The result is an incoherent global map, especially when images are noisy, large slippages occur or loops are closed [22] (for example, see Figure 2.16a). In order to rectify these inconsistencies, we must devise an energy function E that measures the discrepancy between local and global information. By iteratively minimising E we obtain corrected values for the camera and world variables. This minimisation process is known as bundle adjustment: simultaneously refining the feature points that describe the scene structure as well as the viewing parameter estimates, such that they are jointly optimal.

In the bundle adjustment literature, E typically takes the camera's intrinsic/extrinsic parameters and 3D points as arguments [28]. However, given the planar geometry constraint, we develop a more

simplified formulation that can be applied in either 2D or 3D. Let $\mathbf{H}_w^j$ – the third-column-deprived projection matrix from (2.20) – be written as:

$$\mathbf{H}^j = \begin{bmatrix} \mathbf{H}_U^j \\ \hline \mathbf{H}_L^j \end{bmatrix} = \left[\begin{array}{ccc} h_{11}^j & h_{12}^j & h_{13}^j \\ h_{21}^j & h_{22}^j & h_{23}^j \\ \hline h_{31}^j & h_{32}^j & 1 \end{array}\right] \tag{2.30}$$

Alternatively, in the 2D case we can set $\mathbf{H}^j = \mathbf{H}_0^j$. Given a global feature $\hat{\mathbf{f}}_i \in \mathbb{R}^2$, let J_i denote the set of all images which contain a feature that maps to $\hat{\mathbf{f}}_i$. We wish to minimise the Euclidean distance between each global feature, $\hat{\mathbf{f}}_i$, and all local features that map to it, $\mathbf{f}_i^j \ \forall \ j \in J_i$. A simple objective function is one which takes the squared sum of reprojection errors for all features with multiple matches:

$$E = \sum_i \sum_{j \in J_i} \left\| \mathbf{\Omega}(\hat{\mathbf{f}}_i, \mathbf{H}^j) \right\|^2, \text{where } \mathbf{\Omega}(\hat{\mathbf{f}}_i, \mathbf{H}^j) = \mathbf{f}_i^j - \frac{\mathbf{H}_U^j \begin{bmatrix} \hat{\mathbf{f}}_i \\ 1 \end{bmatrix}}{\mathbf{H}_L^j \begin{bmatrix} \hat{\mathbf{f}}_i \\ 1 \end{bmatrix}} \tag{2.31}$$

$\mathbf{\Omega} = \begin{bmatrix} x_i^j & y_i^j \end{bmatrix}^\top$ is a vector describing the x and y distance error between a global and local feature in the image plane, as demonstrated in Figure 2.13. One method to efficiently minimise the energy function is to linearise the error $\mathbf{\Omega}$ using a first-order Taylor series approximation, and only consider small perturbations in $\hat{\mathbf{f}}_i$ and $\mathbf{H}^j$ about an initial guess. This gives:

$$\mathbf{\Omega}(\hat{\mathbf{f}}_i + \delta\hat{\mathbf{f}}_i, \mathbf{H}^j + \delta\mathbf{H}^j) \approx \Omega(\hat{\mathbf{f}}_i, \mathbf{H}^j) + \frac{\partial \mathbf{\Omega}}{(\partial \hat{\mathbf{f}}_i)^\top} \delta\hat{\mathbf{f}}_i + \frac{\partial \mathbf{\Omega}}{(\partial \text{vec}\mathbf{H}^j)^\top} \delta \text{vec}\mathbf{H}^j \tag{2.32}$$

Note that vec$\mathbf{H}^j$ is the vertical stacking of the columns of $\mathbf{H}^j$. We can then write the energy function

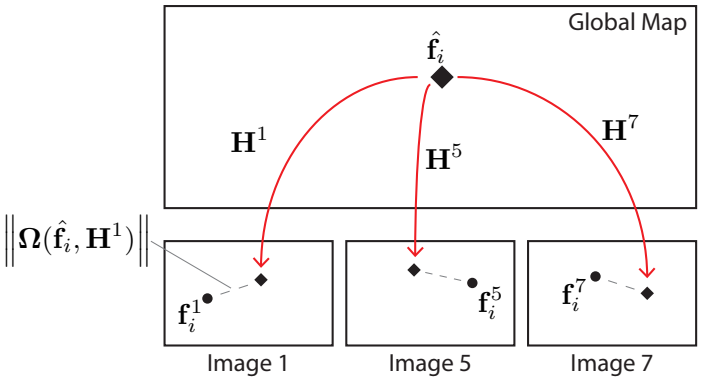


Figure 2.13: A geometric interpretation of the bundle adjustment algorithm. The cost E is computed by squaring and summing the errors $\mathbf{\Omega}$ between observed image features and their corresponding global features reprojected on to the image planes. This is minimised by jointly perturbing the homographies and global features over many iterations.

ALGORITHM 2.1 Loop to build parameters $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and solve optimisation

for all local features with matches
  $[\mathbf{a}_i, \mathbf{b}_i] = \text{get\_optimisation\_params}(\hat{\mathbf{f}}_i, \mathbf{f}_i^j, \mathbf{H}^j)$;
  n_{feat} = feature index locations (2 locations);
  n_{img} = image index locations (8 locations);

  $\mathbf{A}(n_{feat}) \mathrel{+}= \mathbf{a}_i^\top \mathbf{a}_i$;
  $\mathbf{B}_{local} = 2\mathbf{a}_i^\top \mathbf{b}_i^\top$; *// 10×1 vector*
  $\mathbf{B}(n_{feat}) \mathrel{+}= \mathbf{B}_{local}(1:2)$;
  $\mathbf{B}(n_{img}) \mathrel{+}= \mathbf{B}_{local}(3:10)$;
  $\mathbf{C}_{local} = \mathbf{b}_i \mathbf{b}_i^\top$; *// 10×10 matrix*
  $\mathbf{C}(n_{feat}, n_{feat}) \mathrel{+}= \mathbf{C}_{local}(1:2, 1:2)$;
  $\mathbf{C}(n_{img}, n_{img}) \mathrel{+}= \mathbf{C}_{local}(3:10, 3:10)$;
  $\mathbf{C}(n_{feat}, n_{img}) \mathrel{+}= \mathbf{C}_{local}(1:2, 3:10)$;
  $\mathbf{C}(n_{img}, n_{feat}) \mathrel{+}= \mathbf{C}_{local}(3:10, 1:2)$;
end

$E_{old} = \mathbf{A}$;
$\boxed{\text{Solve } \hat{\mathbf{X}} = \underset{\mathbf{X}}{\text{argmin}}\ \mathbf{B}^\top \mathbf{X} + \mathbf{X}^\top \mathbf{C} \mathbf{X};}$ *// Solve using quadprog in MATLAB*
$E_{new} = \mathbf{A} + \mathbf{B}^\top \hat{\mathbf{X}} + \hat{\mathbf{X}}^\top \mathbf{C} \hat{\mathbf{X}}$; *// Estimated new energy value (not exact due to linearisation)*

as:

$$E = \sum_{i,j} \left\| \boldsymbol{\Omega}(\hat{\mathbf{f}}_i, \mathbf{H}^j) + \begin{bmatrix} \frac{\partial \boldsymbol{\Omega}}{(\partial \hat{\mathbf{f}}_i)^\top} & \frac{\partial \boldsymbol{\Omega}}{(\partial \text{vec}\mathbf{H}^j)^\top} \end{bmatrix} \begin{bmatrix} \delta \hat{\mathbf{f}}_i \\ \delta \text{vec}\mathbf{H}^j \end{bmatrix} \right\|^2 = \sum_{i,j} \left\| \mathbf{a}_i + \mathbf{b}_i^\top \boldsymbol{\delta}_i^j \right\|^2 \tag{2.33}$$

$$\therefore\ E = \sum_{i,j} (\mathbf{a}_i^\top \mathbf{a}_i + 2\mathbf{a}_i^\top \mathbf{b}_i^\top \boldsymbol{\delta}_i^j + \boldsymbol{\delta}_i^{j\top} \mathbf{b}_i \mathbf{b}_i^\top \boldsymbol{\delta}_i^j), \tag{2.34}$$

where $\|\cdot\|$ is the Euclidean norm, $\mathbf{a}_i$ is a 2×1 constant vector, $\mathbf{b}_i$ is a 10×2 matrix containing the derivatives of $\boldsymbol{\Omega}$ (Jacobians) with respect to all 10 unknown parameters, and $\boldsymbol{\delta}_i^j$ is the 10×1 vector of unknowns. By stacking all unknowns into a single vector $\mathbf{X} = \begin{bmatrix} \delta \hat{\mathbf{f}}_1 & \cdots & \delta \hat{\mathbf{f}}_i & \cdots & \delta \mathbf{H}^1 & \cdots & \delta \mathbf{H}^j & \cdots \end{bmatrix}^\top$, the energy minimization becomes an unconstrained quadratic optimisation problem:

$$\hat{\mathbf{X}} = \underset{\mathbf{X}}{\text{argmin}}\ \mathbf{A} + \mathbf{B}^\top \mathbf{X} + \mathbf{X}^\top \mathbf{C} \mathbf{X} \tag{2.35}$$

For n matched global features and m views there are $k = 2n + 8m$ unknowns. Accordingly, $\mathbf{A}$ and $\mathbf{B}$ are k-vectors and $\mathbf{C}$ is a sparse, real-symmetric $k \times k$ matrix (see Figure 2.15b). Algorithm 2.1 outlines how these global parameters are constructed using $\mathbf{a}_i$ and $\mathbf{b}_i$. In MATLAB, $\mathbf{C}$ is stored as a *sparse* matrix to avoid storing and manipulating the zero blocks. Once $\hat{\mathbf{X}}$ is computed, the homographies and matched feature frames are updated and the adjustment is repeated until convergence or for a fixed number of iterations. Thereafter, global features visible from only one view are updated given the new homographies.

Note that the inverse objective function was also considered, whereby the error between global features and local features projected onto the global plane is measured. The primary drawback of this

formulation is that it creates scaling problems: the cost function can be trivially minimised by down-scaling the reconstructed scene. Tests on a variety of real and synthetic datasets consequently found that this formulation tended to get caught in nonsensical minima and introduce perspective distortions more often. Conversely, measuring the misalignment locally provided improvements in stability and rate of convergence. Specifically, the down-scaling problem is overcome because errors are measured in the image space at a fixed scale.

2.4.1 Performance Evaluation and Improvements

The minimisation of E is an ill-posed problem – that is, its solution is not unique. As such, it must be re-formulated by introducing a set of additional assumptions and constraints on the parameters to ensure that the algorithm converges to a physically sensible solution.

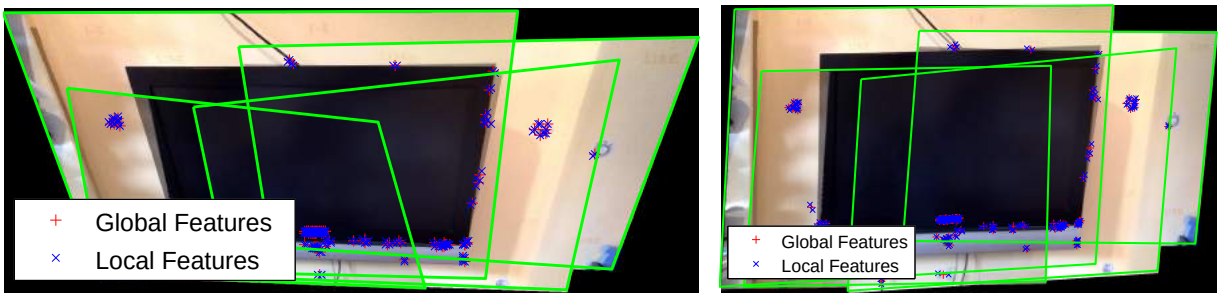
Scaling: The primary cause of instability is the scale invariance of the energy function. This is tied to the issue of scale-depth ambiguity: although the cost cannot be trivially minimised through down-scaling, there still exist an infinite number of structurally similar solutions since homographies and feature positions are only defined up to an unknown factor. Accordingly, a constraint must be implemented that approximately fixes the scale of images with respect to the reference. Let us write the transformation in (2.30) as:

$$\mathbf{H}^j = \left[\begin{array}{cc} \mathbf{A} & \mathbf{t} \\ \hline \mathbf{p}^\top & 1 \end{array}\right] \tag{2.36}$$

It can be shown that for an inhomogeneous linear transformation from $\mathbb{R}^2 \to \mathbb{R}^2$ (i.e. the case when $\mathbf{p} = \mathbf{0}$ and $\mathbf{H}^j$ is affine), $|\det \mathbf{A}|$ is equal to the scale factor by which area is multiplied under the mapping [1]. Furthermore, the sign of the determinant indicates whether orientation is preserved, so we should find that $\det \mathbf{A} > 1$ assuming that images are not mirrored onto the world or reference plane. The affinity assumption is also reasonable for a downward-facing camera that is roughly fronto-parallel to the ground plane. Thus, under the notion that the perspective effect is negligible, we can preserve scale by imposing the following constraint:

$$W(\mathbf{A}) = (\alpha \det \mathbf{A} - 1)^2 \approx 0 \tag{2.37}$$

$\alpha = 1$ in the 2D case where transformations from the reference image are being optimised ($\mathbf{H}^j = \mathbf{H}_0^j$); however, when transformations from the world plane are considered ($\mathbf{H}^j = \mathbf{H}_w^j$) we must ignore the scale change between the world and reference image, hence $\alpha = \left|\det \mathbf{A}_w^0\right|$. Considering small perturbations in accordance with the bundle adjustment algorithm, we can approximate (2.37) as a



(a) *Before regularisation – significant distortion* **(b)** *Perspective penalty introduced* ($\lambda = 10^8$)

FIGURE 2.14: Bundle adjustment tends to introduce perspective distortion when there are few matched features (red crosses above indicate global features registered from multiple views). Regularisation helps to counteract this issue: the parameter λ governs the extent to which affinity is enforced in the homographies.

linear constraint set below a threshold value t:

$$W(\mathbf{A} + \delta\mathbf{A}) \approx W(\mathbf{A}) + 2\left(\alpha^2 \det \mathbf{A} - \alpha\right) \frac{d(\det \mathbf{A})}{d(\mathrm{vec}\, \mathbf{A})} \delta \mathrm{vec}\mathbf{A} \leq t, \tag{2.38}$$

$$\text{where } \frac{d(\det \mathbf{A})}{d(\mathrm{vec}\, \mathbf{A})} = \mathrm{vec}\left(\mathrm{adj}\, \mathbf{A}^\top\right)^\top \tag{2.39}$$

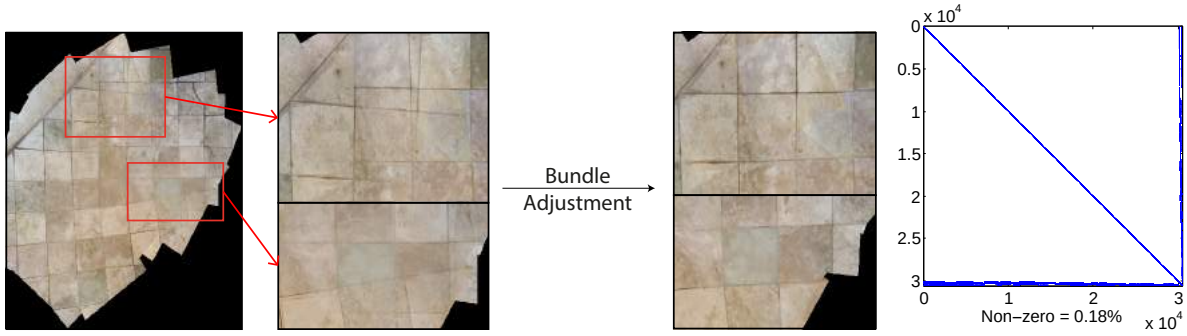
In the MATLAB application a value of $t = 1$ was used.

Perspective Distortion: Another degeneracy that became apparent after implementing the scaling constraint was that bundle adjustment introduced significant and somewhat erratic perspective distortion in the mosaic. The effect is unsurprising given our assumption that photos are taken approximately fronto-parallel to the ground. It was particularly pronounced when the feature map was sparse and the link graph contained few images and no loops (see Figure 2.14a). This is overcome by introducing a term in the energy minimisation that penalises large perspective values $\mathbf{p}$ – a technique known as regularisation. Thus, the objective function is comprised of the quadratic form as in (2.35) and a regularising functional:

$$E_{reg} = E + \lambda \sum_j \left[(h_{31}^j + \delta h_{31}^j)^2 + (h_{32}^j + \delta h_{32}^j)^2 \right], \tag{2.40}$$

where h_{31}^j, h_{32}^j are as defined in (2.30) and $\lambda > 0$ is a suitably chosen regularisation parameter. Introducing this term produced the improvements shown in Figure 2.14b.

Other Improvements: For larger maps it was found that an alternation approach, in which homographies and feature points were adjusted independently, reduced computational cost whilst maintaining fast convergence when large errors existed. The linearised objective function is greatly simplified since the derivative terms in (2.32) for the fixed set of unknowns (which alternate between $\mathbf{f}$ and $\mathbf{H}$



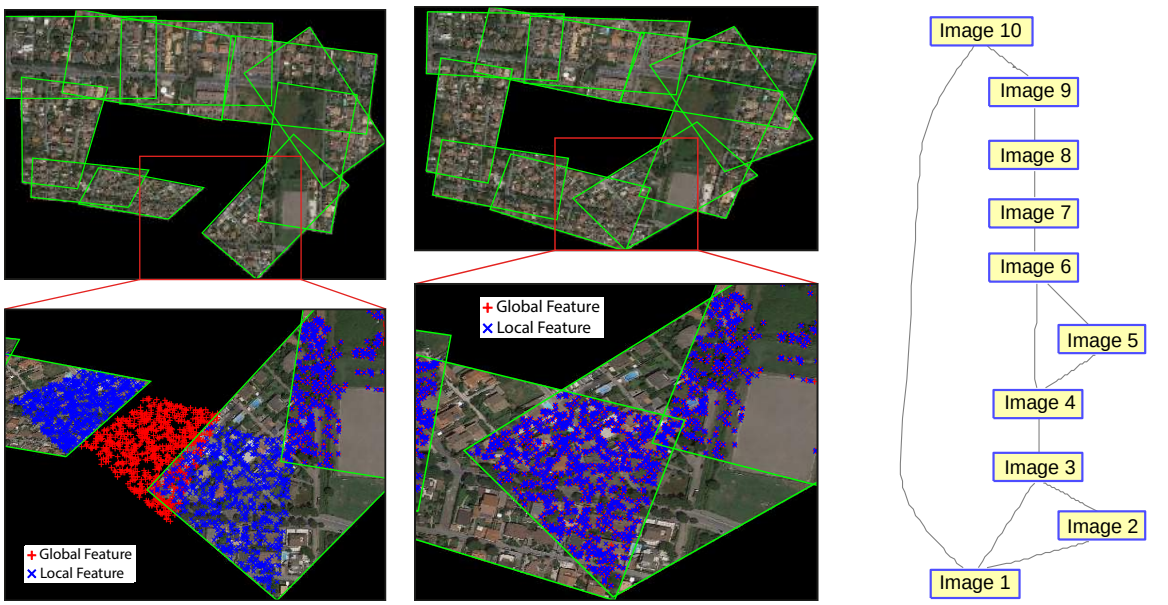
(a) *Significant improvement in detail after bundle adjustment* **(b)** *Sparsity pattern of* $\mathbf{C}$

FIGURE 2.15: The algorithm produces visually pleasing results when making granular refinements in feature-dense scenarios. The above example shows a mosaic of 54 photos with 30,542 unknown parameters ($\mathbf{X} \in \mathbb{R}^{30542}$). Before, there are large errors due to perspective distortion and drift. The algorithm adjusts global feature positions $\hat{\mathbf{f}}_i$ and image transformations $\mathbf{H}^j$ jointly to minimise the objective function. $\mathbf{C}$ (right) forms an "arrowhead" matrix, such that individual features lie along the top-left $2n \times 2n$ block diagonal and are coupled to images in the last $8m$ rows/columns if their corresponding elements are non-zero.

at each iteration) reduce to zero. This observation is consistent with other work and has spawned a class of "alternation algorithms". However, this approach tends to ignore the coupling between cameras and features (governed by the off-diagonal blocks of $\mathbf{C}$), thus neglecting that the cost of a feature displacement can be partly offset by a compensatory camera displacement and vice versa [28]. Furthermore, its computational benefits arise primarily from updating homographies independently rather than features, since we find that the number of views is much smaller than the number of points (hence $8m \ll 2n$). Consequently, visually and computationally optimal results were obtained when using "full" bundle adjustment and optimising only the homographies every 10-30 iterations. This method was found to be particularly effective when correcting errors accumulated over large loops, as in Figure 2.16.

Overall, the final implementation produces promising results. Figure 2.15 demonstrates its ability to make fine adjustments to a real dataset. Figure 2.16 illustrates the elegant handling of the loop closure problem for a mosaic created using screenshots from Google Earth: the topological map registers loops by considering local matches between images, while the bundle adjustment module ensures that these loops are geometrically enforced and globally consistent.

Moving toward larger-scale and offline applications demands further revisions. The algorithm scales poorly as the system of simultaneous equations, generally solved in polynomial time by quadratic programming, grows rapidly with the number of images. Furthermore, convergence is relatively slow compared to the state-of-the-art, which is somewhat characteristic of first order methods (although this often comes at reduced computational cost) [28]. One solution is to divide-and-conquer by partitioning the data into several sets, bundle adjusting each one independently and then merging them. Mouragnon et al. [17] have successfully implemented this top-down model in real time by considering reprojection



(a) *Before bundle adjustment* **(b)** *After bundle adjustment* **(c)** *Correspondence graph*

FIGURE 2.16: Bundle adjustment robustly closes loops. Figure 2.16a shows that in the initial mosaic, created by cascading frame-to-frame transformations, differences in perspective distortion between images results in a visible gap where the images should meet. After bundle adjustment, the loop is properly closed and differences in perspective distortion are reconciled. The algorithm was run for 500 iterations and energy reduced by a factor of about 2000.

errors in only the r most recent key-frames – a system termed "Local Bundle Adjustment". Other potential improvements include data trimming of outliers for increased robustness (which could be achieved through further regularisation), exploiting the sparsity of $\mathbf{C}$ by some method of factorisation, and preconditioning, which reduces the sensitivity of gradient descent methods to the co-ordinate system [28].

Chapter 3: Localisation

Having established a mechanism for generating a two-layered map, the next step is to localise a new view within it. At the outset, this may seem trivial: we could simply follow the querying process described in Section 2.1.2 by searching through indexed images, filtering out matches and computing the global transformation from pairwise homographies. Although feasible, this approach has a number of significant drawbacks:

- It neglects the adjusted features in the global map and instead uses local information – in fact, only the topological map is utilised.
- New information gathered from query images cannot be used to inform the accuracy of existing observations in the feature map.
- Indexed images could include occlusions or partial views, such that no individual image fully encompasses the query object. This creates inaccuracies during spatial verification.
- Inefficiencies are introduced if there are many redundant views of the same object.

We instead develop a baseline system that searches the entire feature map as a single virtual image. By merging data into a common co-ordinate system and making use of bundle adjusted SIFT frames, a global feature-based search scheme maximises the availability of globally aligned features for geometric verification, thus ensuring that the retrieved pose parameters are as accurate as possible. Two revisions are made to the baseline: in Section 3.2, a map partitioning algorithm is devised to address the issue of computational complexity; in Section 3.3, methods to further prune redundant or inaccurate information from the map are considered.

3.1 Baseline System

Figure 3.1 exhibits the preliminary setup. The feature map acts as a single 'virtual image' viewed by an ideal camera (with $\mathbf{K} = \mathbf{I}$) fronto-parallel to the surface, such that its image plane coincides with the world plane. Localisation entails a single, slightly modified geometric verification stage. Image-to-image verification assumed that feature matches were one-to-one: when a single feature (word) in the query image had multiple target matches, only the first correspondence was considered. Due to quantisation, this assumption collapses for a large textured map where repeating structures are likely to exist, so ambiguous matches cannot be discarded. Whereas previously the verification phase had complexity $O(n^2)$ for n matched words in the query image, it now increases to $O(\alpha^2 n^2)$, where α is a branching factor denoting the average number of matches per feature.

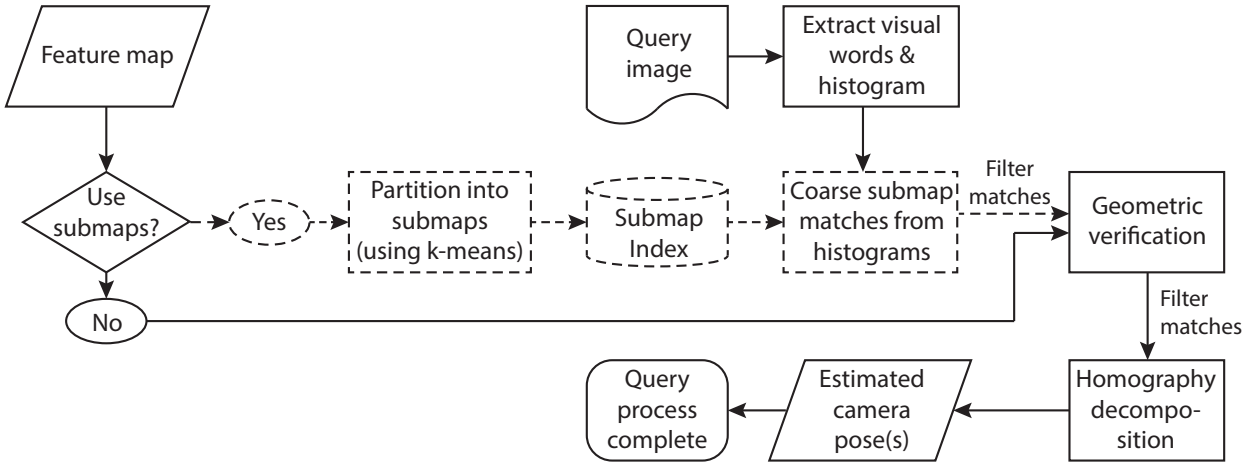
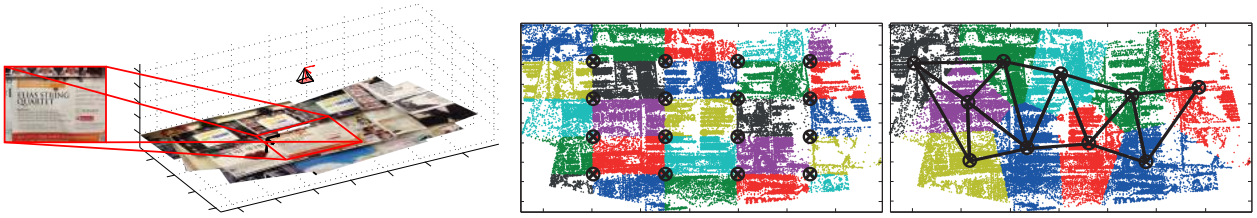


FIGURE 3.1: The localisation process. In the baseline scheme the dashed region is skipped, so the query image is spatially verified against the entire feature map immediately after feature extraction and quantisation. Division of the map into submaps allows an initial filtering stage based on histogram matching scores, which improves speed significantly (see Figure 3.4).



(a) *Localising a notice on a bulletin board* **(b)** *Two alternative submapping schemes for Figure 3.2a*

FIGURE 3.2: Figure 3.2b demonstrates the partitioning of features under the two clustering approaches. For rectangular segments (left), the coloured regions that meet at each vertex (denoted by the crosses) are combined to produce overlapping submaps. For k-means segments (right), each Delaunay triangle joining three Voronoi cells forms an individual submap (generating 9 submaps in the above example).

The baseline produces good results (see Figure 3.2a) at an unsurprisingly high cost. It also places substantial pressure on appropriate selection of the vocabulary size. A smaller dictionary means α is likely to be large due to coarse quantisation, but too many clusters could mean that Voronoi cells are too tight to absorb descriptor noise, resulting in false negative matches.

3.2 Submap-Based Localisation

To counteract the branching factor's computational restriction on scalability, the global map can be partitioned into several virtual images that are queried as described in Chapter 2. This introduces a preliminary histogram matching stage to filter out many of the correspondences that would otherwise contribute to ambiguity (illustrated in Figure 3.3). Consequently, α is reduced for each submap and fewer features are spatially verified overall – an inefficient process compared to histogram-based matching, which by contrast has linear time complexity in the number of submaps.

The rising demands of SLAM have led to the development of comprehensive submapping strategies such as Atlas – a probabilistic and hybrid metric/topological framework [5]; these schemes fully

incorporate the submap model into pose estimation, path planning and optimisation algorithms. Reviewing our existing mapping and bundle adjustment methods to interface seamlessly with submaps is a non-trivial task necessary for real-time applications. However, it is considered beyond the scope of this project, which instead focuses on the performance implications of map partitioning *after* scene reconstruction and bundle adjustment.

The simplest solution is to divide the map into a set of overlapping rectangular areas. For maximum localisation performance, the overlap region should be equal to the expected dimensions of the projected image to ensure that a query image does not lie at the intersection between two submaps. This is easy to estimate if we assume that images are captured from the same height. In practice, the scheme suffers when maps are sparsely distributed – a likely outcome for robots navigating along long paths that leave many rectangular regions empty. Consequently, submaps are not ideally centred about the dense regions of the map, making the choice of submap size difficult without prior knowledge.

To overcome this complication, one approach is to combine adjacent submaps with few features and discard those with none. An alternative method is to divide the 2D feature map into Voronoi cells using k-means clustering. Noting that the Delaunay triangulation is the dual structure of the Voronoi diagram in $\mathbb{R}^2$, overlapping submaps can be constructed by combining cells that form a Delaunay triangle, as illustrated in Figure 3.2b. Since the triangulation joins all Voronoi vertices that share common edges, the Voronoi diagram derived from k-means clustering must be clipped outside of the map range. However, this improved segmentation comes with the significant computational burden of the k-means algorithm. Scalability can be improved using approximate nearest neighbours (ANN) and quality maintained by employing k-means++ initialisation [4] for randomised seeding based on the data points.

Both submapping schemes were implemented in MATLAB and it was found that a single iteration of clustering significantly enhanced the distribution of submaps. For a given number of partitions, localisation performance was similar for small, evenly distributed maps; however, the k-means method expectedly prevailed for larger maps with sparsely distributed features and extended paths or loops. Figure 3.4 demonstrates the staggering decline in computation time with the number of submaps when localising the image in Figure 3.2a. The original map contained approximately 150,000 global features

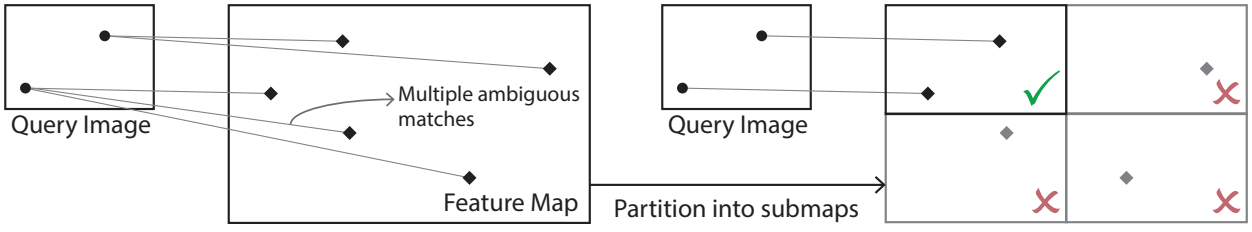
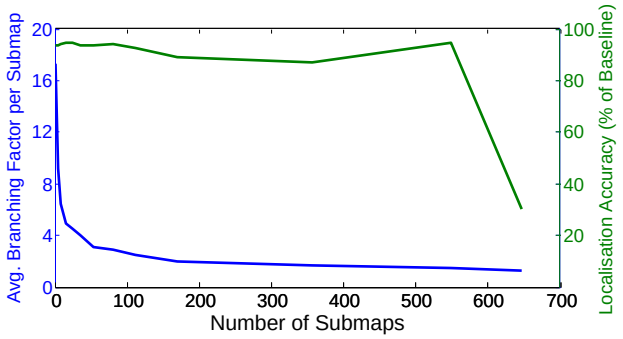
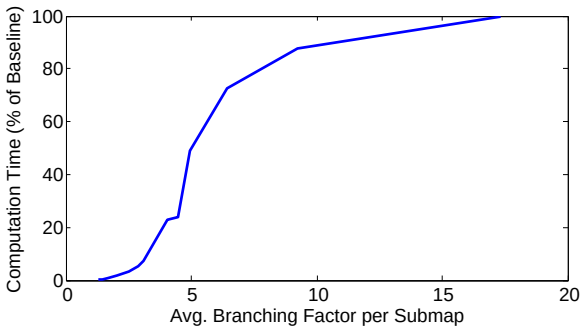


Figure 3.3: Map partitioning introduces performance improvements



(a) *Increasing submaps significantly reduced the branching factor without impacting retrieval performance*



(b) *Time taken to localise the image (match histograms and spatially verify)*

Figure 3.4: The localisation task from Figure 3.2a was performed with varying numbers of submaps. On the left, localisation performance was measured using the overlap criterion described in Chapter 4. Branching factor α reduced by over 90% without significant impact on accuracy. On the right, computation time varies quadratically with α as hypothesised. The slope levels off after values of α that correspond to fewer than 10 submaps because spatial re-ranking was performed on the top 10 matches.

and 10,000 visual words, which justifies the branching factor of 17.3 for the baseline. Somewhat surprisingly, the localisation module was capable of handling 550 submaps (originally constructed from 10 images) with 99.5% reduction in computation time ($\alpha = 1.5$) and no discernible reduction in accuracy. Beyond this point, retrieved homography estimates rapidly degenerated and the system failed. However, generating this many submaps is unwise: it is slow, increases the risk of failure, and introduces rising memory requirements without much incremental improvement in retrieval speed. Moreover, incorporating them in a compatible bundle adjustment module would likely lead to further complexity.

3.3 Data Purging

Incorrect data associations, temporary occlusions and false negative matches create a tendency for the map to grow indefinitely in bounded environments, meaning that submaps become saturated with features. This poses the risk of the system becoming sluggish or degenerating entirely as more images are indexed, and is especially problematic for long-term mapping applications. A typical data association strategy for landmark-based SLAM is to assign a counter C to each global feature [20]; during a query, all spatially verified features are incremented, whilst any unassociated local features are initialised in the world map with $C = 1$. Although unequipped with a probabilistic framework, we can cautiously state that all global features falling within the query image's projection on the world plane are "expected" to be associated. We can therefore assign an analogous penalty value P to each global feature, which is incremented when an expected association does not occur. Features are subsequently purged (or merged to their nearest neighbour if it is sufficiently close) when $P - C$ exceeds a threshold value, thus regulating and stabilising memory consumption of the world map over time.

Cases where C and P are both large may provide evidence for overfitted visual words whose definitions are too restricted to absorb noise in SIFT descriptors. This tends to arise if the vocabulary tree is excessively large relative to the training data. Cummins and Newman [9] address this by modifying k-means to add a cluster merging heuristic. After each k-means iteration, if two centres are closer than a threshold value of similar magnitude to the estimated descriptor noise, one of them is re-initialised at a random location.

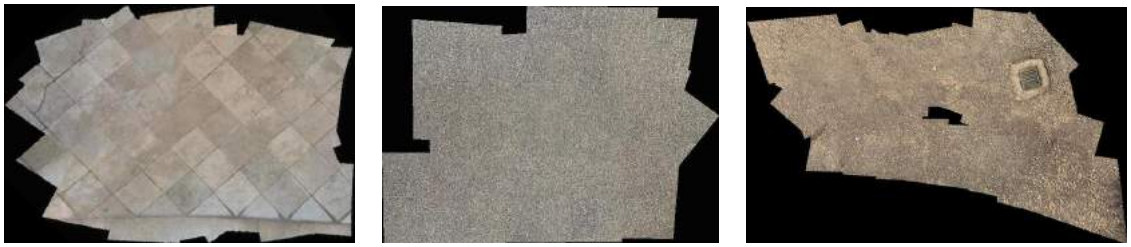
Chapter 4: Data and Evaluation

Having evaluated the performance of bundle adjustment and submapping, we now propose a more regimented testing approach to assess the recognition quality of the localisation system as a whole. One particular challenge is to establish a metric to quantify the localisation performance of test data, given that the feature map is only defined up to scale and may deviate from the ground truth. To reduce the coupling of mapping and localisation errors, the ground truth is instead defined within the feature map space. By using the "overlap criterion" outlined in Section 4.2.1, we can examine the localisation performance in greater isolation. Note that this chapter is not concerned with the accuracy of feature maps, as this was evaluated qualitatively in Section 2.4.1 using visual cues such as parallel lines; a complete numerical evaluation of mapping capability would be cumbersome as it requires direct measurement of the ground truth.

This chapter first introduces the test data (Section 4.1). Methods for assessing the localisation algorithm under different conditions such as lighting, camera angle and camera height, are then outlined (Section 4.2). Finally, results are presented and evaluated, including examples where the system failed (Section 4.3).

4.1 Data

The three datasets used for testing (Figure 4.1) were carefully selected to represent diverse textures. These range from one that is vaguely discernible to the naked eye (marble tiles), to those that a human would have significant difficulty piecing together (carpets). All photos were taken using an iPad Mini (at approximately the same height and fronto-parallel to the ground) and were down-sampled to a resolution of 765 $\times$ 1024 for faster processing. Intrinsic calibration parameters were computed using the *Camera Calibration Toolbox for MATLAB* [7] – a one-off calculation assumed to have negligible errors. The vocabulary size and number of training images used in each dataset are shown in Figure 4.1; both should be sufficiently large that the dictionary generated provides a good representation the different scene descriptors, but without overfitting.



(a) *Marble (54, 100000)* **(b)** *Carpet (20, 15000)* **(c)** *Concrete (36, 50000)*

Figure 4.1: Datasets used for testing with (num_images, num_words) shown

4.2 Testing Methodology

The system's ability to produce the accurate textured maps in Figure 4.1 is alone evidence that localisation works well when images are "safe": undistorted, well-illuminated and taken from the same height. The following tests were designed to deduce the conditions that drive the system to failure. For each of these tests, the overlap criterion was used as the key performance measure (see Section 4.2.1). The Parallel Computing Toolbox in MATLAB was utilised to benefit from the processing power of both cores in the test computer. In particular, the parallel for-loop *parfor* distributes the order-independent task of geometrically verifying the top matches to two MATLAB "workers" that run in parallel (the same technique was also used during the build process for the link graph and feature map). Re-rank depth was set to 5 for all three tests, which are outlined below.

Camera Angle (Perspective): The camera was pivoted about a point to obtain four sets of query images, with the camera's principal axis at approximately 90° (perpendicular), 70°, 50°, and 30° to the ground plane.

Camera Height: Query images capturing same patch were taken from approximately 1/3, 2/3, 1 and 2 times the height of the mapped images (always perpendicular to the ground plane).

Lighting: To mimic different lighting conditions, an individual query image's exposure value can be manually adjusted. On one end the image is overexposed: highlights are blown out (saturated) and shadows are noisy; on the other, the image is underexposed. In both extreme scenarios the dynamic range of the image declines, losing detail in areas of very high or low intensity. Testing across a range of exposures should reveal the robustness of feature extraction/quantisation and image localisation to varying lighting intensities. However, this artificial technique does not account for the noise effect in low light conditions: cameras must compensate for poor lighting with longer shutter speeds, which cause salt-and-pepper noise, or higher sensor gain (ISO sensitivity), which can be modelled by additive white Gaussian noise. These two effects are also investigated on underexposed query images.

Regular point-and-shoot cameras must judge appropriate aperture and shutter speed values by measuring the intensity of reflected light entering the sensor – a process known as *metering*. Their

Figure 4.2: The same query image post-processed to produce five exposures ranging from –3EV to +3EV

light meter adjusts the scene such that pixel intensities average to some fixed (middle grey) value. Consequently, cameras tend to underexpose when photographing predominantly white or reflective surfaces and overexpose dark surfaces. The lighting test therefore implicitly assesses the stability of the localisation algorithm when cameras misjudge exposure in this manner. For the purposes of our experiment, exposure values (EV) varied from –3EV to +3EV in increments of 1.5EV, as shown in Figure 4.2.

4.2.1 Overlap Criterion

The "overlap criterion" formulated here is a performance metric used to value the accuracy of localisation. The user manually selects four points on the mosaic corresponding to corners of the query image to approximate the ground truth. Let the 2D area on the world plane contained within this "true" image projection be denoted by A; the area within the retrieved (estimated) image projection is B. We thus define:

$$O = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

This is illustrated in Figure 4.3 and is computed numerically by setting the number of steps in the x or y direction. An important feature of O its scale invariance, as it is measured in percentage terms. At each camera angle, height and exposure level, overlap values were computed for five query images. Only the *top* returned locations were considered, with their means and standard errors reported in Section 4.3.

There are two methods by which we can compute B. The first is to use the estimated projective homography $\mathbf{H}$ obtained directly from spatial verification. This is entirely independent of the parameters $\mathbf{R}$ and $\mathbf{t}$ obtained from homography decomposition and ignores the constraint that $\mathbf{H}$ should take the form of a roto-translation. The resulting overlap values thus measure the accuracy of 2D matching

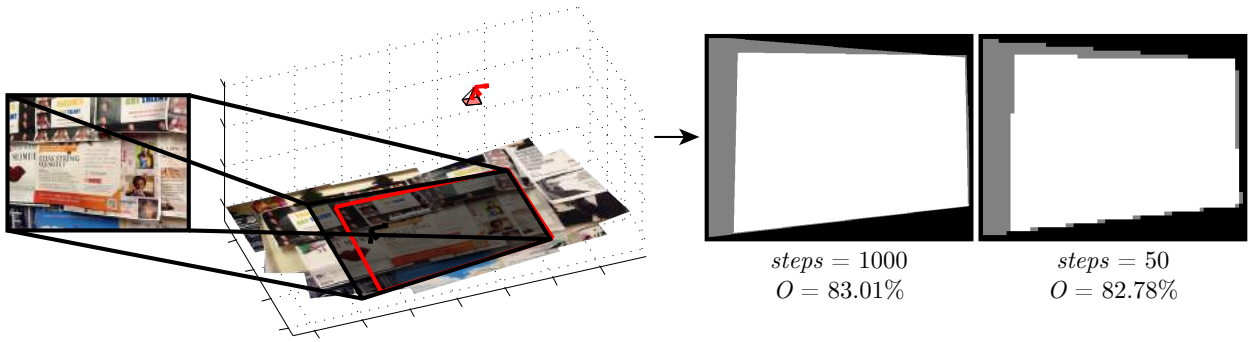


Figure 4.3: For simplicity, overlap is computed numerically. The "true" corners of the image on the mosaic, as selected by the user, are shown by the shaded area. The red area is the query image projection estimated by the localisation module. On the right, the white area is the intersection of the ground truth and estimate $(A \cap B)$, whereas the grey area is $(A \cup B) \cap \overline{(A \cap B)}$.

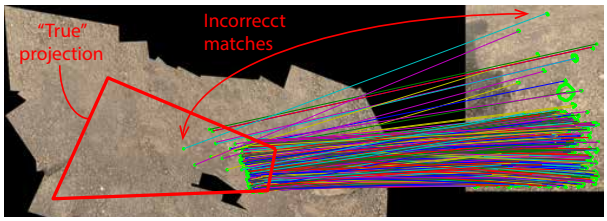


Figure 4.4: Concrete performed poorly at large camera angles (left) due to incorrect matching of distant features

Table 4.1: Variation in overlap performance with camera angle. Note that values for the carpet dataset are given as probabilities of successful localisation $p_{cor} = \mathrm{E}[\epsilon]$, as defined in Section 4.2.1.

Angle (°):	30-40	50-60	70-80	90
Marble (O)	88±3	90±2	95±1	96±1
Concrete (O)	58±5 [59±6]	71±4	76±4	96±0
Carpet (p_{cor})	0.6±0.1 [1.0±0.0]	1.0±0.0	1.0±0.0	1.0±0.0

and spatial verification. Conversely, one could reconstruct $\mathbf{H}$ from $\mathbf{R}$ and $\mathbf{t}$, giving a value of O that is dependent on the estimated 3D camera pose (and intrinsic matrix $\mathbf{K}$) without requiring direct measurement of 3D ground truth. In fact, by examining the *difference* in overlap values between these two methods, we gain insight into the magnitude of error introduced by homography decomposition. In our tests, we begin by evaluating 2D localisation capability and later discuss additional errors introduced when estimating 3D pose.

Note that the textured scenes being tested are, by their very definition, difficult to discern by the naked eye. Consequently, the human input process to obtain ground truths is very slow, requiring close examination of the mosaics. Repeated trials found that human error generally accounted for about ±2-3% variation in O, and further systematic error is likely introduced by distortions in the map. However, for the carpet texture it was difficult to make any more than an educated guess of the ground truth based on prior knowledge of where photos were taken. Accordingly, results for this dataset do not explicitly report overlap values (which alone hold little value), but instead the probability of correct localisation p_{cor} (where "correct" is defined as $O > 0.5$). This is computed as follows: for each of the five query images in a test ($q = 1, \ldots, 5$), the binary variable $\epsilon_q \in \{0, 1\}$ is set to 1 if $O_q > 0.5$. Then $p_{cor} = \frac{1}{5}\sum_q \epsilon_q$ is simply the mean of these results.

4.3 Results

The tables presented here document the mean and standard error in O and p_{cor}, each derived from five query images. In light of the recorded errors, our results serve more to identify broad directional trends rather than fine differences. Table 4.1 demonstrates the sensitivity of homography estimates to variations in camera angle. The saliency of tile edges ensured that the marble dataset maintained good accuracy throughout the range; by contrast, the carpet and concrete scenes did not have any outstanding structural features, hence performance declined significantly at larger (more slanted) angles. Figure 4.4 reveals the effect taking place at these extremes: distant features become blurred and too small for detection, typically because the iPad's fixed aperture is too large to keep far-away objects in

Height (%):	1/3	2/3	1	2
Marble (O)	91±1	95±1	96±1	95±1
Concrete (O)	91±1	93±1	96±0	82±3 [82±3]
Carpet (p_{cor})	1.0±0.0	1.0±0.0	1.0±0.0	0.2±0.1 [0.8±0.1]

(a)

Exposure (EV):	–3	–1.5	0	+1.5	+3
Marble (O)	0 [72±18]	93±2	97±1	92±1	0 [59±17]
Concrete (O)	96±1	96±1	96±1	96±1	96±1
Carpet (p_{cor})	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0

(b)

Table 4.2: Overlap values obtained when varying camera height (left) and query image exposure levels (right)

focus (known as a shallow *depth of field*). By neglecting or mismatching distant keypoints, matched features are not well distributed across the query image and the system *underestimates* the perspective effect.

One means of reducing this error is to decrease the scale of the first octave in the Gaussian scale space to detect smaller features. The scale space is comprised of a set of images that are progressively smoothed by a Gaussian kernel, which equates to a successive reduction in the image resolution. By searching over several octaves, the SIFT detector identifies keypoints at multiple scales, thus achieving scale invariance. The results in square brackets in Tables 4.1, 4.2 and 4.3 indicate the overlap attained when starting the scale space with a smoothing level of 0.5 instead of 1 (roughly equivalent to doubling the query image resolution), such that smaller features could be detected. Tests found that this technique improved robustness for the carpet dataset (which contained many fine structures), but was generally ineffective with concrete. It also tended to increase keypoint extraction time by a factor of 2 to 3.

Table 4.2a shows that localisation generally remained accurate over a wide range of camera heights, with marble performing best as before. Where failure occurred for the carpet due to a decline in keypoint scale, this was often rectified using the double resolution technique discussed above. Surprisingly, both concrete and carpet were very resilient to changes in lighting (Table 4.2b), whereas localisation failed entirely at the extreme exposures for marble – perhaps an indication that smooth textures are more sensitive to uniform changes in lighting. Once again this was occasionally corrected by modifying the scale space, although the approach has proven somewhat inconsistent across different textures.

The response of the system to noise that arises in low-light conditions is very promising. Table 4.3 shows how overlap varied with the density D_N of salt-and-pepper noise (S&P) and the variance σ_N^2 of additive white Gaussian noise (AWGN) when applied to underexposed query images (at –1.5EV). The concrete dataset was tolerant to very high levels of both types of noise, and despite their relatively weaker performance the marble and carpet textures were also reliable at levels of noise well above those that arise in practice. Reducing the smoothing kernel was actually found to *harm* results, as the system would confuse random noise for genuine features. Figure 4.5 demonstrates the remarkable amount of noise applied to images that were successfully localised, *just below* the failure level. In all

S&P (D_N):	5%	10%	20%	50%	60%
Marble (O)	91±2	89±2	0 [0]	–	–
Concrete (O)	96±1	96±1	96±1	96±1	0 [0]
Carpet (p_{cor})	1.0±0.0	1.0±0.0	0.2±0.1 [0]	–	–

AWGN (σ_N^2):	0.02	0.05	0.10	0.40	0.60	0.70
Marble (O)	90±2	86±3	0 [0]	–	–	–
Concrete (O)	96±1	96±1	96±1	96±1	94±2	18±40 [0]
Carpet (p_{cor})	1.0±0.0	1.0±0.0	0.2±0.1 [0]	–	–	–

Table 4.3: Overlap values when artificially applying salt-and-pepper (left) and additive white Gaussian noise (right).

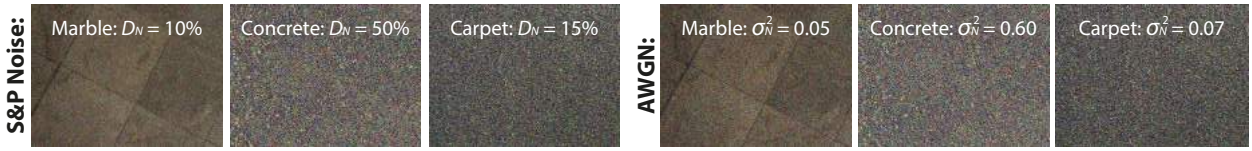


Figure 4.5: The maximum levels of noise at which query images were correctly localised

cases, the scheme unequivocally outperforms the naked eye.

Having assessed only the 2D aspect of localisation (keypoint matching and geometric verification) we now observe the errors introduced from 3D pose estimation (homography decomposition into a roto-translation). Table 4.4 compares the original overlap values with those obtained when projecting the query image using the re-composed homography from $(\mathbf{R}, \mathbf{t})$. It is evident that extension from 2D to 3D recovery introduces small projection errors due to physical constraints imposed on the camera motion and inaccuracies in $\mathbf{K}$. The impact is negligible when the camera is approximately fronto-parallel to the ground plane, but at larger camera angles overlap is more materially affected. This is due in part to the fact that O is more sensitive to small changes in 3D pose at these extremes, as the image projection elongates more dramatically along the ground plane at slanted angles. Volatility in O also manifests itself in the form of higher standard errors at large angles (Tables 4.4 and 4.1), indicating that performance becomes more variable at these levels.

There are a multitude of other configurable parameters that impact the detection of SIFT features, such as the descriptor measurement region and octave resolution (number of scale levels sampled per octave). However, tweaking of these values did not provide any notable improvements and often introduced additional computational overhead.

Angle (°):	30-40	50-60	70-80	90
Marble (H)	88±3	90±2	95±1	96±1
Marble ($\mathbf{R}, \mathbf{t}$)	88±3	90±2	94±1	96±1
Concrete (H)	58±5	71±4	76±4	96±0
Concrete ($\mathbf{R}, \mathbf{t}$)	50±6	58±5	65±4	96±0

Height (%):	1/3	2/3	1	2
Marble (H)	91±1	95±1	96±1	95±1
Marble ($\mathbf{R}, \mathbf{t}$)	86±2	89±1	96±1	95±1
Concrete (H)	91±1	93±1	96±0	82±3
Concrete ($\mathbf{R}, \mathbf{t}$)	90±1	87±1	90±1	79±3

Table 4.4: Overlap values when using the transformation $\mathbf{H}$ obtained directly from spatial verification versus the homography constructed from $\mathbf{R}$ and $\mathbf{t}$.

Chapter 5: Conclusion

This project has sought to design and develop a comprehensive, scalable localisation and mapping framework for textured surfaces. A complete MATLAB implementation has been developed as proof of concept, building on the state-of-the-art in object retrieval by utilising the VLFeat and visualindex libraries [29]. All of the features documented in this report have been incorporated in the application, which is available for download online[1]. A shift to production code would require conversion to a compiled language for optimum efficiency (C would be ideal since it interfaces with VLFeat). The specification in Section 1.3.1 listed several desirable attributes of the system, each of which is reflected upon here.

Texture and Matching/Data Association: *Robustly estimate position in self-similar planar environments.* In many respects this primary objective was achieved: the system demonstrated rapid and consistent localisation capability in scenes which even the naked eye could not distinguish without painstaking examination. Tests in a variety of environments (Section 4.3) exhibited the algorithm's resilience toward changes in lighting, scale, perspective and noise, sometimes further improved by reducing the minimum Gaussian smoothing level. The kidnapped robot problem is also elegantly overcome. Much of this performance can be attributed to the robustness of SIFT descriptors, and provides evidence that quantised words retain the invariance of original descriptors when the vocabulary is well trained. However, although tests revealed the system's ability to achieve consistent overlap values of 90% and over in divergent settings, they hinged on relatively few data points as ground truth estimates required costly human input.

Loop Closure: *Actively identify when known locations are re-visited and incorporate this information into a global map.* The interplay between the two layers of map abstraction – the link graph and feature map – gracefully manages loop closures (see Sections 2.2.1, 2.2.2 and 2.4.1). The link graph maintains high level image-to-image correspondences whilst the feature map encapsulates their more granular physical geometry, with each layer working independently.

Optimisation: *Globally correct the map such that historical data (accumulated features and homographies) are consistent.* The bundle adjustment algorithm devised in Section 2.4 exploits the dual structure of the link graph and feature map to ensure that the two are consistent. Further enhancements implemented in Section 2.4.1, such as scale constraints and regularisation, significantly improve stability, preventing the algorithm from descending into physically nonsensical minima. The optimisa-

[1] Download code from: https://github.com/jaijuneja/texture-localisation-matlab

tion was shown to perform well on real and synthetic datasets and corrected loop closures. However, it also leaves much to be desired in terms of computational complexity and only scales to about 100 images in MATLAB.

Scalability: *Operate smoothly over large environments.* The bag-of-words model offers a very solid basis for up-scaling. However, as it does not carry spatial information a computationally laborious geometric verification step was required. To address this bottleneck, a submapping scheme was developed in Section 3.2, delivering substantial computational improvements over the baseline system without impacting localisation.

5.1 Further Work

Several avenues for further development in data purging and bundle adjustment were discussed in Sections 3.3 and 2.4.1. For instance, a more sophisticated approach to data purging and merging could be achieved by employing a probabilistic model for classifying correspondences. Classes of matches could include those that fit a geometric model (either planar or more general), those that derive from moving objects or temporary occlusions, and those that are false.

Pose estimation could also benefit from a probabilistic framework. Earlier tests only accounted for the top match, whilst discarding all other information. An improved scheme might combine the retrieved values of $\mathbf{R}$ and $\mathbf{t}$ using uncertainty information such as matching scores.

The proposed system has great capacity for real-time applications such as mobile robotics. In this context, the link graph and feature map would lend themselves well to a path planning module that uses graph search and local geometric information to deduce the optimal path between waypoints. Computational complexity could also be reduced by searching only nearby submaps. However, the current system only operates offline and is crippled by a slow bundle adjustment algorithm compared to the state-of-the-art – an issue that was investigated with proposed modifications in Section 2.4.1.

Applications such as large crowd-sourced projects might utilise an unlimited number of cameras with unknown internal parameters. Hence, the issue of calibration would become non-trivial and solutions to auto-calibration would need to be considered.

Finally, it may also prove fruitful to study the scope of textural image matching in other domains. For example, our favourable results suggest that a computer may be more capable of distinguishing an authentic painting from a replica. This has exciting implications for the use of visual data to create digital signatures.

References

[1] Determinants and linear transformtions. `http://mathinsight.org/determinant_geometric_properties`.

[2] A. Angeli, S. Doncieux, J. Meyer, U. Pierre, M. C. Paris, and D. Filliat. Real-time visual loop-closure detection. In *in IEEE International Conference on Robotics and Automation (ICRA)*, pages 1842–1847, 2008.

[3] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[4] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[5] M. Bosse, P. Newman, J. Leonard, and S. Teller. SLAM in large-scale cyclic environments using the atlas framework. *International Journal of Robotics Research*, pages 1113–1139, 2004.

[6] T. Botterill, S. Mills, and R. Green. Bag-of-words-driven single camera simultaneous localisation and mapping. *Journal of Field Robotics*, 2010.

[7] J. Y. Bouguet. Camera calibration toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, 2008.

[8] L. A. Clemente, A. J. Davison, I. D. Reid, J. Neira, and J. D. Tardós. Mapping large loops with a single hand-held camera. In *Robotics: Science and Systems*. The MIT Press, 2007. ISBN 978-0-262-52484-1.

[9] 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.

[10] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067, June 2007. ISSN 0162-8828.

[11] E. D. Eade and T. W. Drummond. Unified loop closing and recovery for real time monocular SLAM. In *Proc. BMVC*, pages 6.1–6.10, 2008. ISBN 1-901725-36-7. doi:10.5244/C.22.6.

[12] F. Fraundorfer, C. Engels, and D. Nistér. Topological mapping, localization and navigation using image collections. In *IROS*, pages 3872–3877. IEEE, 2007.

[13] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691.

[15] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003. ISBN 0387008934.

[16] E. Malis and M. Vargas. Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, INRIA, 2007.

[17] E. Mouragnon, Maxime Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 363–370. IEEE Computer Society, 2006.

[18] D. Nistér and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2161–2168. IEEE Computer Society, 2006. ISBN 0-7695-2597-0.

[19] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[20] S. Riisgaard and M. R. Blas. SLAM for dummies: A tutorial approach to simultaneous localization and mapping. Technical report, 2005.

[21] R. Rosenholtz. General-purpose localization of textured image regions. In Michael J. Kearns, Sara A. Solla, and David A. Cohn, editors, *NIPS*, pages 817–823. The MIT Press, 1998. ISBN 0-262-11245-0.

[22] S. Se, D. G. Lowe, and J. J. Little. Vision-based mapping with backward correction. In *IROS*, pages 153–158, 2002.

[23] G. Simon and M. Berger. Pose estimation for planar structures. *IEEE Computer Graphics and Applications*, 22(6):46–53, 2002.

[24] G. Simon, A. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *Proc. International Symposium on Augmented Reality*, pages 120–128, October 2000.

[25] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[26] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[27] N. Tomatis and I. Nourbakhsh. Simultaneous localization and map building: A global topological model with local metric maps. In *In Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 421–426, 2001.

[28] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, pages 298–375. Springer Verlag, 2000.

[29] A. Vedaldi and B. Fulkerson. VLFeat - an open and portable library of computer vision algorithms. In *ACM International Conference on Multimedia*, 2010.

[30] C. Xu, B. Kuipers, and A. Murarka. 3d pose estimation for planes. In *ICCV Workshop on 3D Representation for Recognition (3dRR-09)*, 2009.

[31] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, November 2000.