Beyond The Naked Eye: Localisation and Mapping of Textured Scenes

By Jai Juneja (Supervisor: Dr Andrea Vedaldi)

The Premise

The world abounds with seemingly "boring" or "uninformative" visual data.

To the naked eye, these images are **unstructured**: multiple instances of the same texture cannot be distinguished.

However, at a finer scale the textures are highly **unique** and capture a lot of hidden information. This could:

- Be used to localise robots in environments that are physically bare, but texturally rich
- Complement non-vision based localisation and mapping systems

Could a computer be more capable than humans at identifying, localising and piecing together textured scenes?

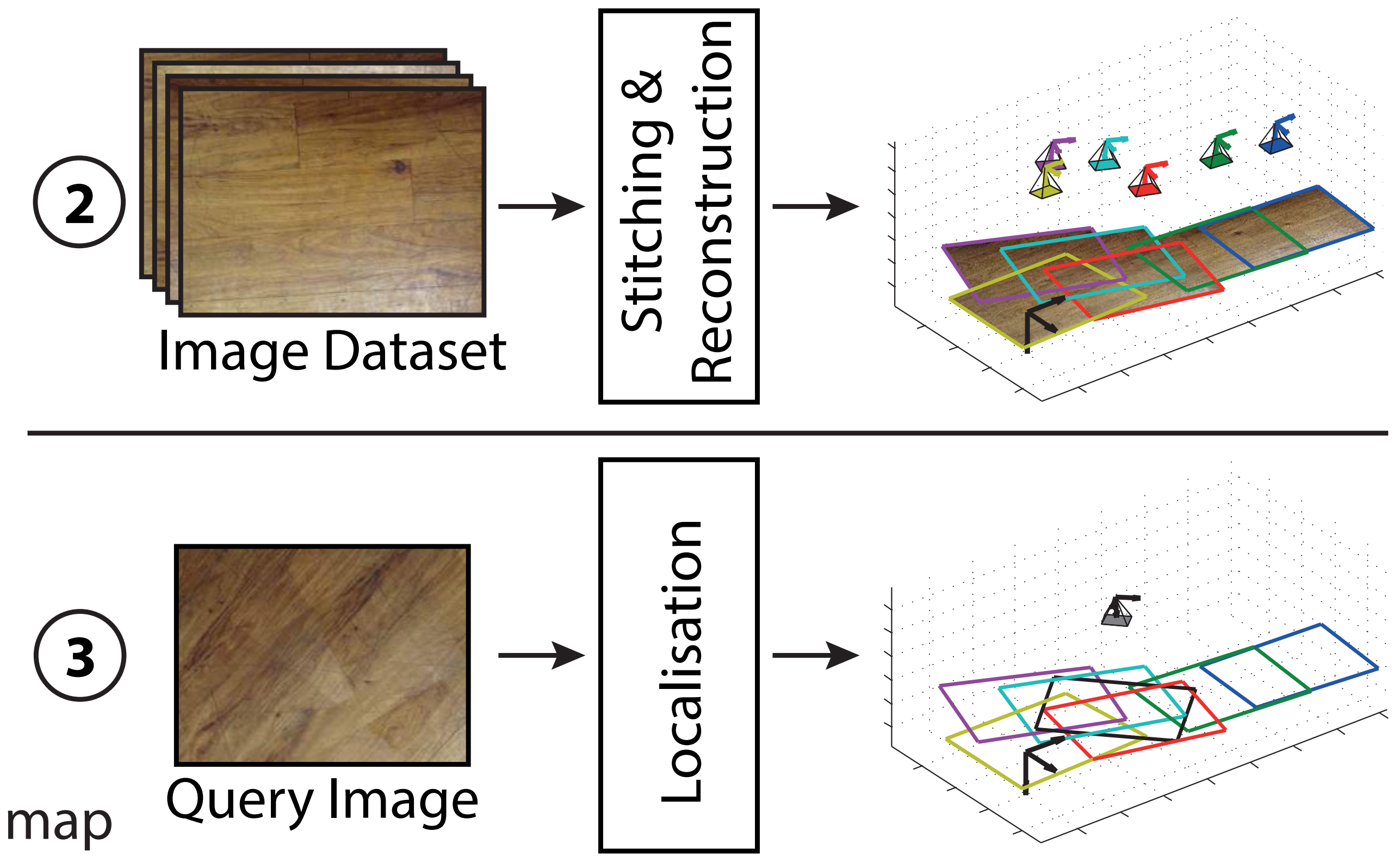
Objectives & Proposed Solution

The system should:

- Reliably estimate camera position in self-similar environments
- Be robust to variations in scale, viewpoint, lighting and noise
- Identify when known locations are revisited - i.e. "close loops"
- Scale to large environments (thousands or millions of images)

Proposed system divided into three modules (see figure on right):

1. **Indexing**: extract image features and efficiently store in memory
2. **Stitching & reconstruction**: get geometric relationships between all images and represent as a global map
3. **Localisation**: get 3D pose of a query image by rapidly searching the map



Step 1: Image Indexing

Use the "**bag-of-visual-words**"-based system described by Philbin *et al.* (2007):

- 128D features extracted from images compacted into single-integer "visual words"

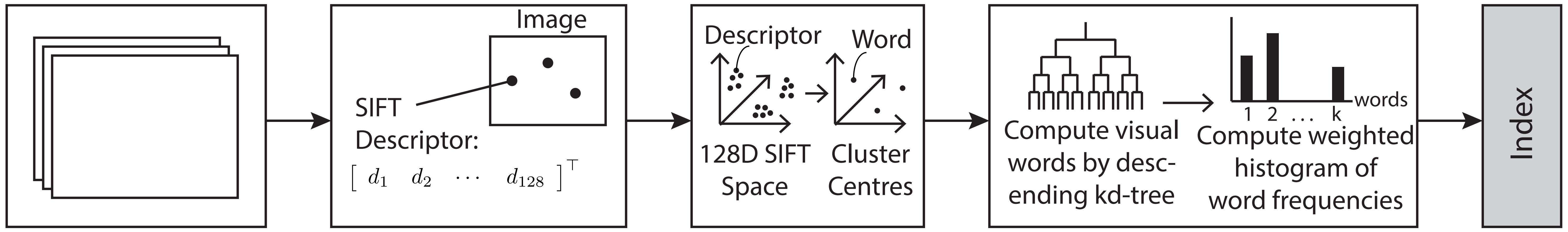


Image
SIFT Descriptor:
$[\ d_1 \quad d_2 \quad \cdots \quad d_{128}\ ]^\top$
Descriptor
Word
128D SIFT Space
Cluster Centres
1 2 ... k
words
Compute visual words by descending kd-tree
Compute weighted histogram of word frequencies
Index
Training Images

Feature Extraction: detect & describe "interest points" (e.g. blobs, edges, corners) using the Scale Invariant Feature Transform [Lowe04]

Vector Quantisation: generate vocabulary (kd-tree) by clustering SIFT vectors into "words" [Sivic03]

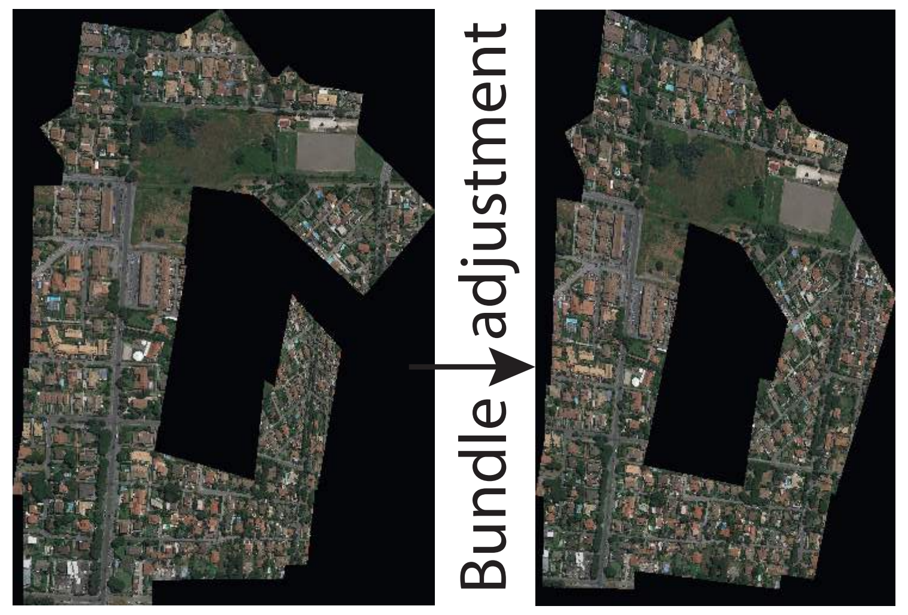
Inverted File Indexing: each image is now represented as a histogram of word frequencies, stored in an "inverted file" for rapid (highly scalable) search and retrieval

Results

Mapping Performance:

Drift errors accumulated from cascading image-to-image homographies:

- Errors were reduced & global consistency enforced through bundle adjustment
- Algorithm performed well over large loop closures (see satellite image dataset, right)



Localisation Performance:

Robust to a variety of textures, e.g.:

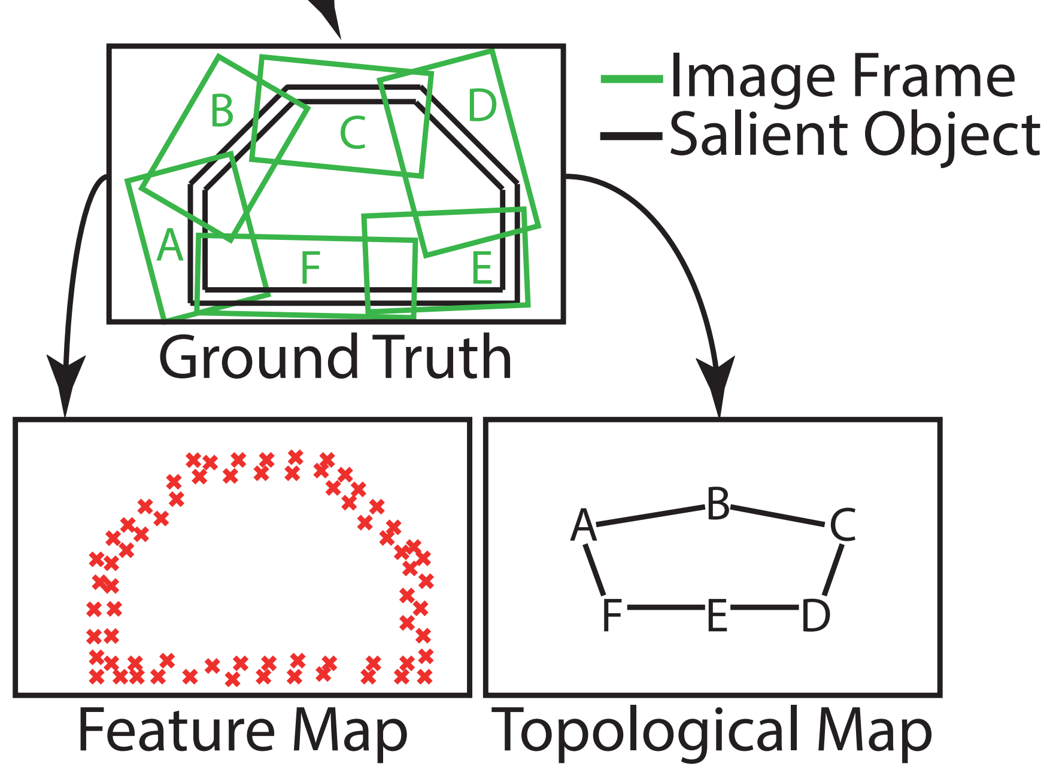
Marble (54 images)

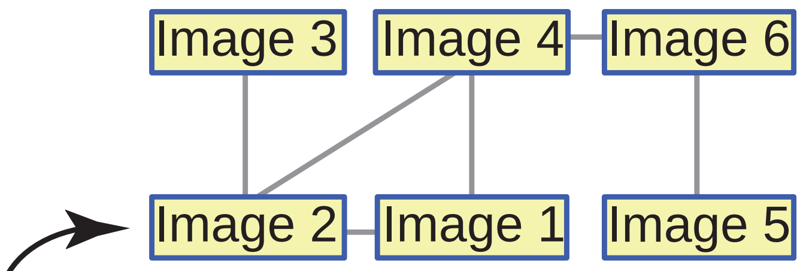
Concrete (36 images)

Carpet (20 images)

Step 2: Stitching & Reconstruction

World is represented by a **two-layered map** (below):



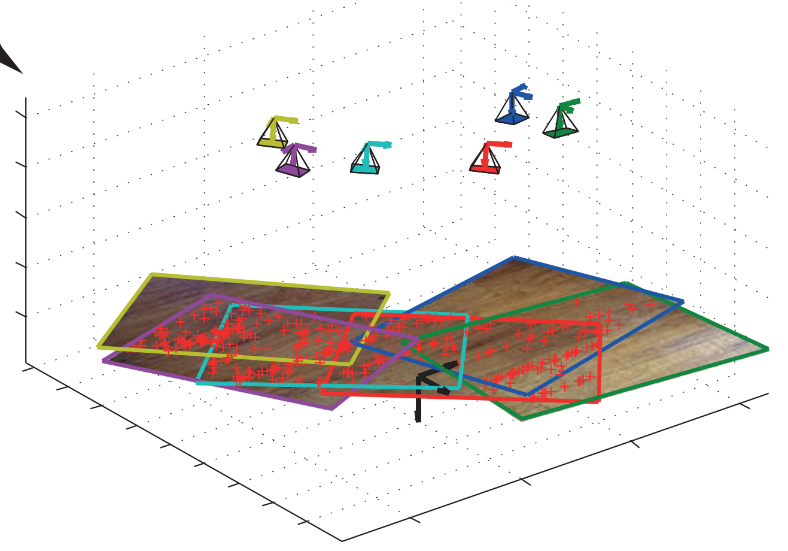


Topological map (link graph) maintains high-level relationships between images:

1. Each image represented by graph node
2. Each image queried against index by matching word histograms & ranking matches by score
3. Transformations between top matches estimated by "spatial verification" [Philbin07]
4. Overlapping images deduced & corresponding nodes connected

Feature map manages finer physical geometry in the scene:

1. Arbitrary reference image chosen
2. All transformations to world plane deduced by cascading pairwise homographies from the reference image
3. All local SIFT features transformed to world plane
4. 3D camera poses (rotation & translation) computed by decomposing image-to-world homographies [Simon02]
5. "Bundle adjustment" jointly optimises camera poses and feature positions in the global map



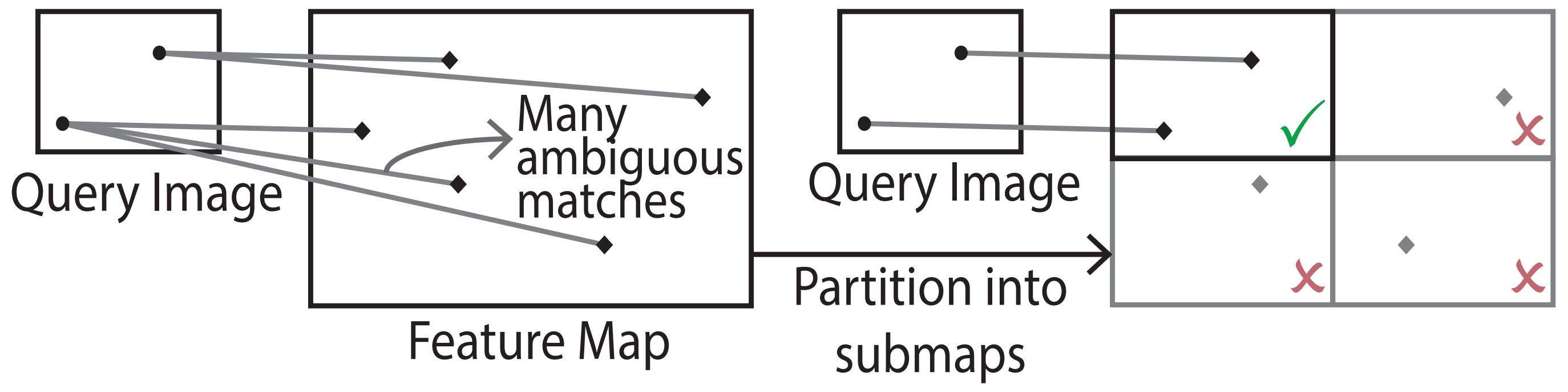
Step 3: Localisation

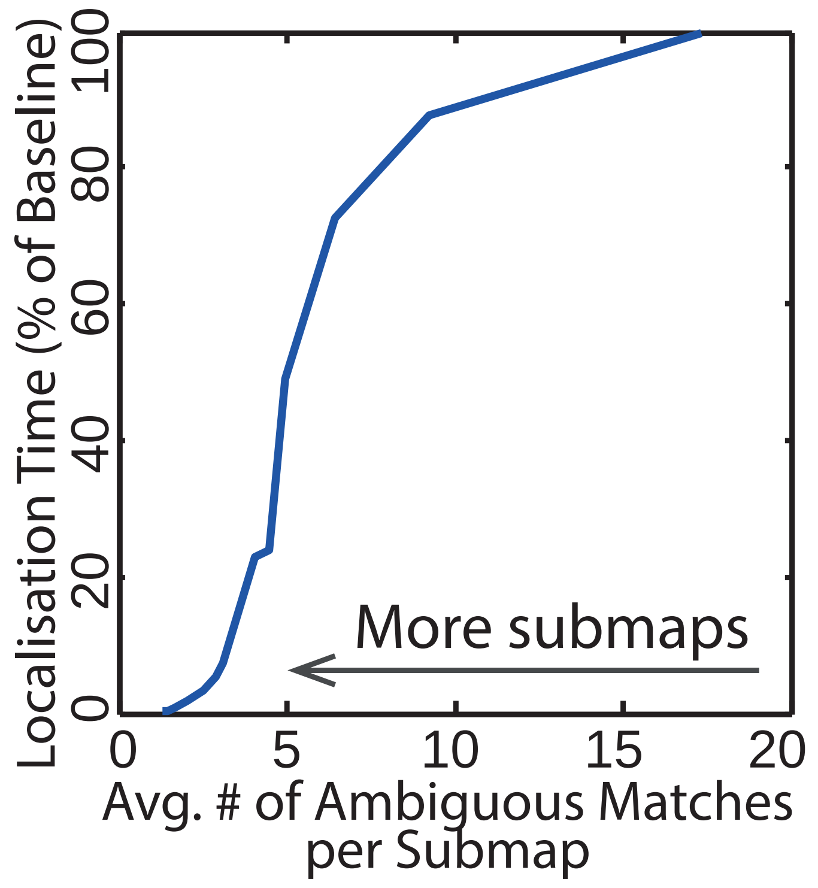
Baseline system localises query images by performing spatial verification against entire global map:

- Homography from image to map estimated using random selection of feature matches
- Number of inlier matches counted & process repeated for several iterations
- Refined homography estimate computed using the maximum number of inlier matches
- Very slow because each feature in query image has multiple matches in global map

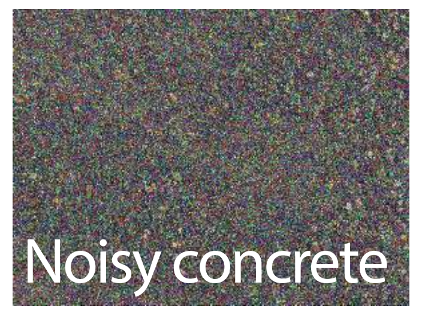
Baseline **speed improved** by partitioning map into 'virtual images', a.k.a **submaps** (see figure below):

- Fast histogram matching step filters out most submaps
- Slow spatial verification step only applied to top matches
- Fewer ambiguous feature matches in each submap dramatically improves performance (see right)





Examples of images that were robustly localised (<10% error):


Noisy concrete

Angled marble

Low-light carpet

Close-up concrete