Bag-of-Words-Driven, Single-Camera Simultaneous Localization and Mapping

Tom Botterill, Steven Mills, and Richard Green

Presented by Sotirios Ch. Diamantas

February 08, 2013

C-MANTIC Lab
Department of Computer Science
University of Nebraska, Omaha

Overview

- This paper describes BoWSLAM, a scheme for a robot to reliably navigate and map previously unknown environments, in real time, using monocular vision alone.

- BoWSLAM can navigate challenging dynamic and self-similar environments and can recover from gross errors.

- BoWSLAM is demonstrated mapping a 25-min, 2.5-km trajectory through a challenging and dynamic outdoor environment without any other sensor input, considerably farther than previous single-camera simultaneous localization and mapping (SLAM) schemes.

Background Work

- A single camera is an ideal sensor for this as it is inexpensive, passive, compact, and non-platform specific and requires relatively little power.

- However, existing schemes for navigating using monocular vision are prone to failure due to gross errors or disorientation in difficult environments.

Background Work

- Data association problem.

- An additional problem with monocular vision is a global scale ambiguity: the actual scale of the world and speed of the camera can be estimated only by identifying something of known size, and small errors in scale can accumulate over time.

- Loop closing problem.

VSLAM and Visual Odometry

- Extended Kalman filter (EKF; Dissanayake,Durrant-Whyte,& Bailey, 2000) or a particle filter (Montemerlo, Thrun, Koller, &Wegbreit, 2002). Successful on small scale.

- Visual odometry is a slightly simpler problem than SLAM in that only the camera motion is of interest and no global map is built. Less computation resources but no loop closure.

VSLAM and Visual Odometry

- There are two major limitations of these approaches, however:
  - first, the environments explored are largely static, and rapid cornering is largely avoided.
  - The second limitation of many single-camera SLAM schemes is their high algorithmic complexity—the SLAM algorithms soon become too costly for real-time operation as the mapped area grows.

The Bag-of-Words Algorithm in Visual Navigation

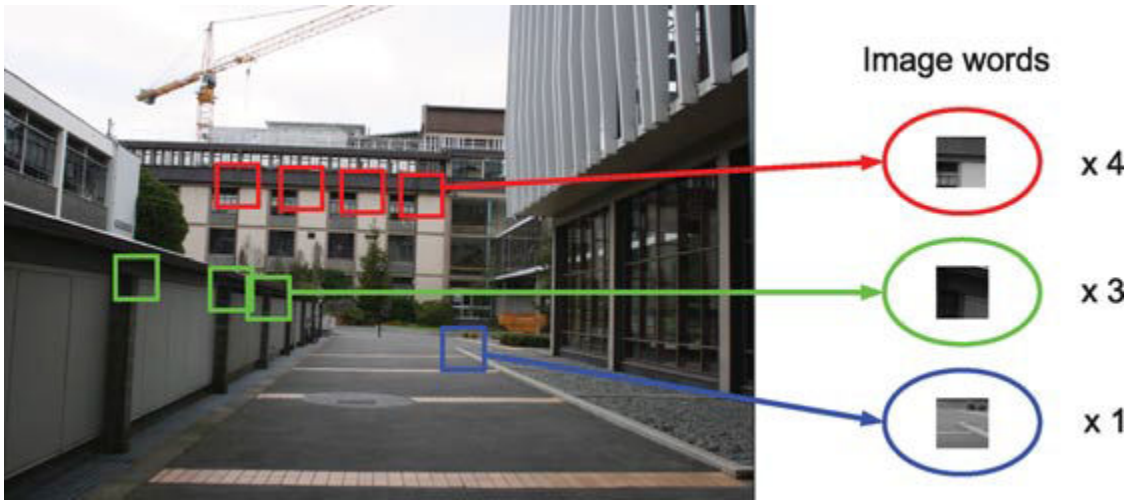
- The BoW image representation describes an image by the set of features that it contains. A set of feature descriptors is extracted from each image, and each descriptor is mapped to its closest match from a quantized “dictionary”.

- The image BoW algorithm was developed for fast and accurate image search.

The Bag-of-Words Algorithm in Visual Navigation

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

The Bag-of-Words Algorithm in Visual Navigation



- To represent an image as a BoW, each descriptor is mapped to the nearest image word.

Requirements for BoWSLAM

- Active loop-closure detection.
  - To recover from gross errors, and to maintain a consistent and accurate map for extended periods, the robot must detect when known locations are revisited, without any prior knowledge of its position
- Wide-baseline matching.
  - The capability to register frames captured from significantly different viewpoints (occlusion, motion blur, or erratic camera motion).
- Robust Position estimation.
  - Position estimation must be robust in the presence of moving objects and in self-similar environments.
- Multiple position hypotheses.
  - Despite robust position estimation, errors will inevitably still occur. Essential for long-term navigation.

BoWSLAM

- FAST corner detector.
- Given a correct set of matches between frames, the relative pose of the camera is computed via a least-squares approach.
- BaySAC framework.

BoWSLAM

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

    -Update graph to select best pose estimate for every frame

Relative Camera Pose Computation

- At least five correspondences are needed to estimate *E*
- Given exactly five correspondences, there are up to 10 possibilities for *E*

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

- *E* is uniquely determined by eight correspondences and can be computed from the matrix *F* that minimizes

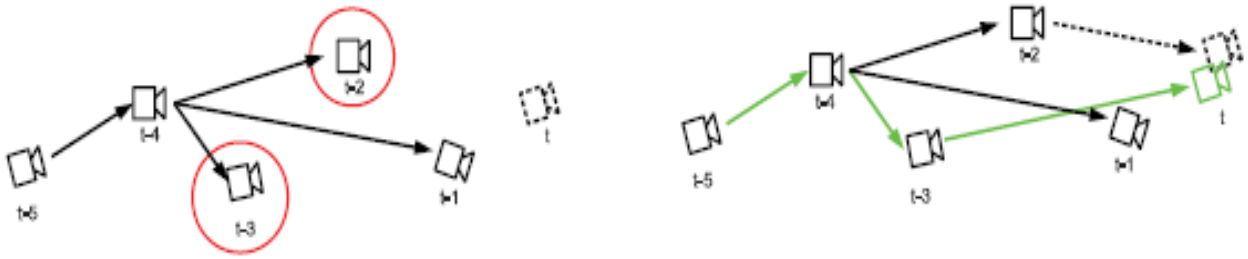
$$\sum_{i=1}^{C} \mathbf{p}_i'^T F \mathbf{p}_i$$

Resolving Scale Change and Multiple Position Hypotheses

- The change in scale between two frames is modeled with a lognormal distribution, $\delta \sim$ Log-$N(d, g2)$ [defined by $\log \delta \sim N(d, g2)$].

- Graph-Based Representation of Multiple Position Hypotheses
  - select frames with many features in common with the current frame. From these candidate frames, choose those that already have accurate position estimates.

Multiple Position Hypotheses



- Camera positions form a graph with edges representing relative position hypotheses and nodes representing frames.

Outdoor Test Results

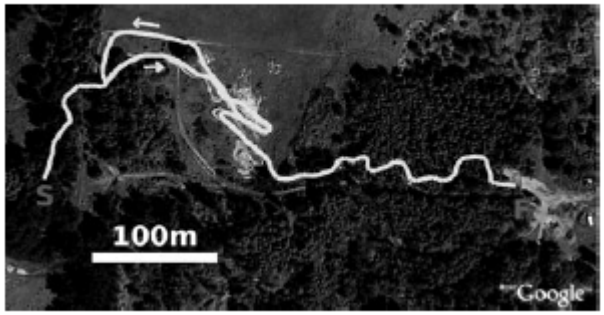
Table I. RMS errors in global maps optimized by TORO.

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

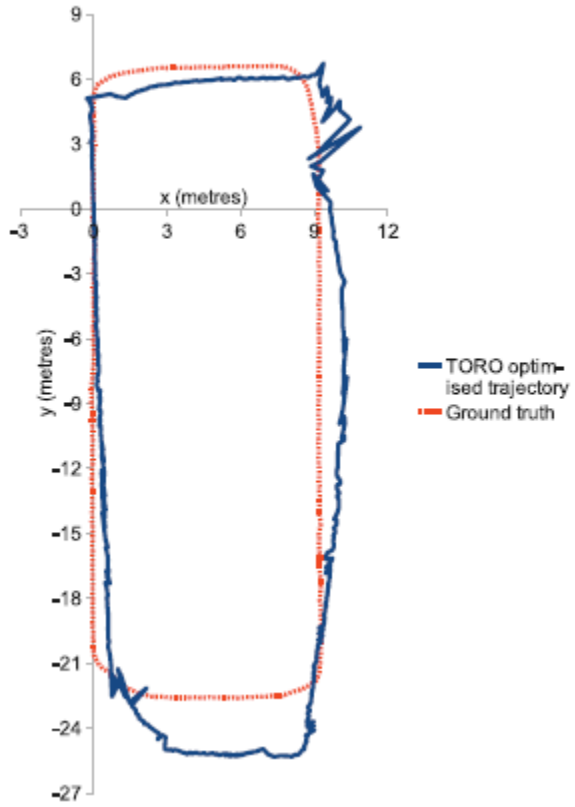
Outdoor Test Results

Outdoor Test Results

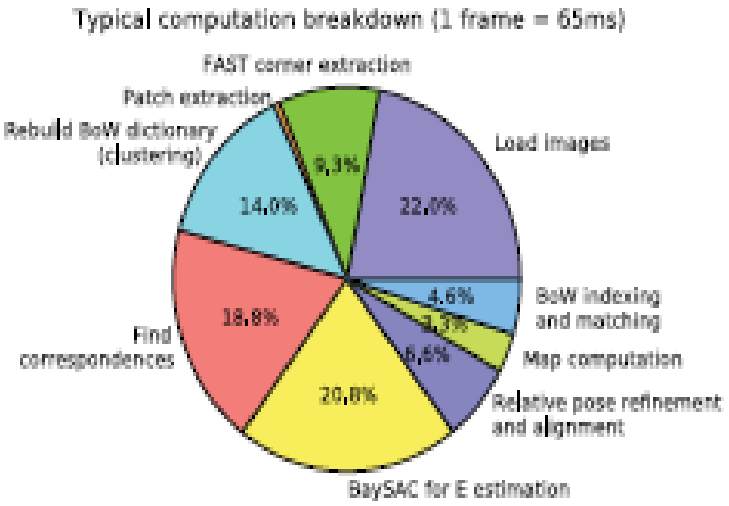
Outdoor Test Results



Indoor Test Results



Results


Typical computation breakdown (1 frame = 65ms)
FAST corner extraction
Patch extraction
Rebuild BoW dictionary (clustering)
Load images
9,3%
14,0%
22,0%
4,6%
BoW indexing and matching
18,8%
Find correspondences
1,3%
Map computation
6,4%
20,8%
Relative pose refinement and alignment
RaySAC for E estimation

Conclusions

- Three innovations make:
  - The use of the BoW algorithm, together with BaySAC, for fast, widebaseline feature matching.
  - The BoW algorithm is used to select good candidate frames from which to compute positions; this allows positioning to continue despite sequences of frames being unusable, for example, due to moving objects.
  - Third, a graph-based representation of multiple position hypotheses allows subsets of good position estimates to be found, despite the presence of gross outliers.

Thank you

Questions?