Snake-SLAM (Prototype)

Darius Rückert
University of Erlangen-Nuremberg

August 30, 2019

Abstract

Snake-SLAM is a sparse, keypoint-based Visual SLAM system for RGBD and monocular input streams. For each frame, ORB features are computed and matched in a multi-stage approach to previously triangulated points. If a new keyframe is required, the best frame out of a history buffer is selected. Bad pose estimates are then refined using the newly created keyframe. Snake-SLAM supports map optimization, large scale loop closure, and relocalization after lost tracking.

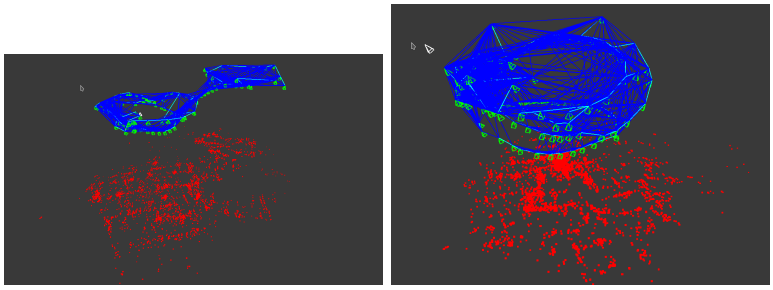

Figure 1: Keyframes and world points for the C3 and C7 test dataset.

1 Initialization

For monocular SLAM systems, a stable initialization is crucial for accurate camera tracking. In many cases, a late initialization with good feature matches is better than a premature initialization with small parallax. In our system, we first compute two-view feature matches and filter them by checking the ratio between the best and second best match. Then, the relative transformation between the images is computed using the 5-point algorithm. The two-view reconstruction is accepted, if it satisfies the following conditions

- 5-point inliers $> th_{in}$

- triangulation angle $\alpha > th_\alpha$
- histogram density $> th_{his}$

2 Tracking

The localization of a novel frame is achieved by matching the image features to previously triangulated points from a predicted camera position. Instead of using all mappoints, only a small subset is selected and used for matching. This subset is called *local map*. If the tracking using the initial local map is successful, the matching is repeated on a second local map which includes more points. This two-stage tracking allows us to use different outlier thresholds and generally improves stability on unpredictable camera motion. If the local map tracking fails, we try to recover it using a two-view reconstruction with the last keyframe. If that fails as well, the system switches into relocalization mode.

During relocalization new frames are matched to all keyframes using a binary bag of word database (see ORB-SLAM). Candidates from the database are then used in a two-view reconstruction to estimate the frame's pose.

3 Keyframe insertion

During exploration, new keyframes must be inserted into the map, because previously unseen geometry might become visible. The traditional approach is to select a frame as a keyframe, if the tracking is *weak*. Weak is very loosely defined here and usually implemented using heuristics. We also use a heuristic to decide when new keyframes are needed, but do not immediately select the latest valid frame as a keyframe. Instead, we keep track of the latest N frames in a history buffer and generate a keyframe from the best candidate. After insertion, all frames after the new keyframe are optimized again.

This method allows us to generate a map of higher quality because better keyframes are selected. For example, if a bad frame with motion blur is tracked, other systems would use that bad frame as a keyframe and degrade the map. Our approach selects a good frame from the past instead.

4 Structure

Snake-SLAM consists of multiple modules which each run independently on different threads (see Figure xx). The input module reads the camera data and computes feature points. The tracking module computes the current pose and decides if a new keyframe should be inserted. Local Mapping, inserts the new keyframe into the map and generates new 3D points. The optimization module, improves the current map by applying bundle adjustment and removing outliers. Loop Closing detects large scale loops and closes them using pose graph optimization.

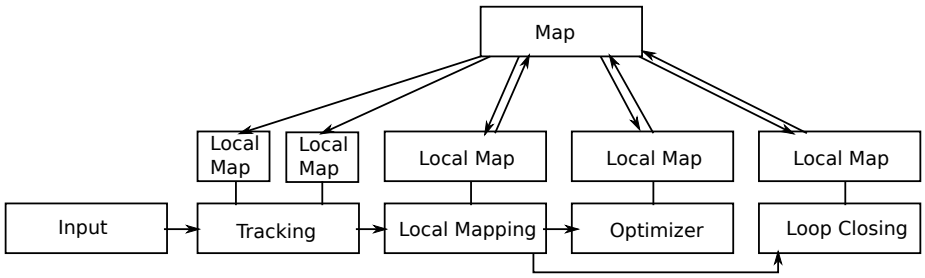

Figure 2: Snake-SLAM overview. Each module operates on local copies of the map.

In contrast to other system (e.g. ORB-SLAM), our modules do not operate directly on the global map. Each module has a local copy of the global map in a local coordinate system. That means, poses compute by the tracking are first transformed into the respective coordinate system before they can be used by the other modules. This was mainly implemented for efficiency reasons. First, the local copies are more cache-friendly, because they only contain the absolutely necessary information. And second, we can get rid of all synchronization primitives except the copy operation between the local and global map.

5 Performance

Currently all modules except the feature detection are well optimized and perform significantly better than ORB-SLAM. The feature detection is work in progress as we want to move it to the GPU. For the ISMAR SLAM challenge we manually capped the frame rate to 100 fps, compiled with debug information, and run all modules sequential on a single thread, because more than 30 fps will not be rewarded. If we enable all optimization and enable multi threading our system can easily reach 800 fps and more on a modern desktop PC.

Module	Average Time (ms)
Feature Detection (CPU)	6
Feature Detection (GPU expected)	< 1
Tracking (1 Thread)	1.1
Tracking (2 Threads)	0.7
Keyframe Insertion	5
Optimization	16
Loop Closing	11

Table 1: Execution time of the individual modules.