ORB-SLAM2

ORB-SLAM2 is a real-time SLAM library for **Monocular**, **Stereo** and **RGB-D** cameras that computes the camera trajectory and a sparse 3D reconstruction (in the stereo and RGB-D case with true scale). It is able to detect loops and re-localize the camera in real time. We also provide a ROS node to process live monocular, stereo or RGB-D streams.
The library can be compiled without ROS.

1. System overview

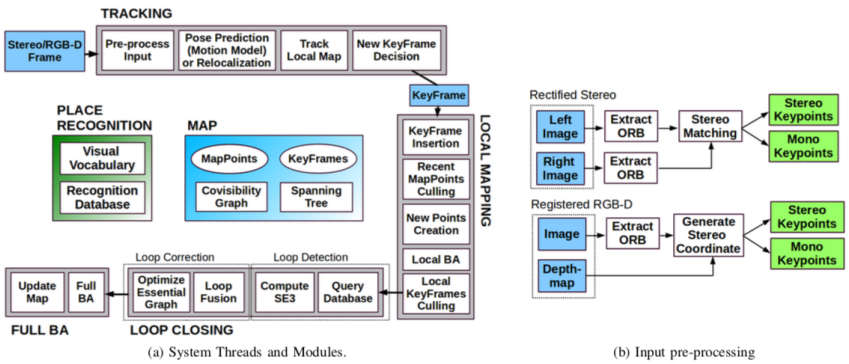
ORB-SLAM2 uses depth information to synthesize a stereo coordinate for extracted features on the image. This way our system is agnostic of the input being stereo or RGB-D. Therefore, our method is lightweight and works with standard CPUs. Our goal is long-term and globally consistent localization instead of building the most detailed dense reconstruction. However, from the highly accurate keyframe poses one could fuse depth maps and get accurate reconstruction on-the-fly in a local area or post-process the depth maps from all keyframes after a full Bundle Adjustment and get an accurate 3D model of the whole scene.

ORB-SLAM2 for stereo and RGB-D cameras is built on our monocular feature-based ORB-SLAM, whose main components are summarized here for reader convenience. A general overview of the system is shown in Fig. 2. The system has three main parallel threads:

1) the **tracking** to localize the camera with every frame by finding feature matches to the local map and minimizing the reprojection error applying motion-only BA.

2) the **local mapping** to manage the local map and optimize it, performing local BA.

3) the **loop closing** to detect large loops and correct the accumulated drift by performing a pose-graph optimization. This thread launches a fourth thread to perform full BA after the pose-graph optimization, to compute the optimal structure and motion solution.



(a) System Threads and Modules. (b) Input pre-processing

Fig. 2. ORB-SLAM2 is composed of three main parallel threads: tracking, local mapping and loop closing, which can create a fourth thread to perform full BA after a loop closure. The tracking thread pre-processes the stereo or RGB-D input so that the rest of the system operates independently of the input sensor. Although it is not shown in this figure, ORB-SLAM2 also works with a monocular input as in [1].

A. Monocular, Close Stereo and Far Stereo Keypoints

ORB-SLAM2 as a feature-based method pre-processes the input to extract features at moving keypoint locations, as shown in Fig. 2b. The input images are then discarded and all system operations are based on these features, so that the system is independent of the sensor being stereo or RGB-D. Our system handles monocular and stereo keypoints, which are further classified as close or far.

Stereo keypoints are defined by three coordinates the coordinates on the left image and the horizontal coordinate in the right image. For stereo cameras, we extract ORB in both images and for every left ORB we search for a match in the right image. This can be done very efficiently assuming stereo rectified images, so that epipolar lines are horizontal. We then generate the stereo keypoint with the coordinates of the left ORB and the horizontal coordinate of the right match, which is subpixel refined by patch correlation. For RGB-D cameras, we extract ORB features on the RGB image and for each feature with the coordinates on the left image we transform its depth value into a virtual right coordinate.

The uncertainty of the depth sensor is represented by the uncertainty of the virtual right coordinate. In this way, features from stereo and RGB-D input are handled equally by the rest of the system.

A stereo keypoint is classified as close if its associated depth is less than 40 times the stereo/RGB-D baseline, otherwise it is classified as far. Close keypoints can be safely triangulated from one frame as depth is accurately estimated and provide scale, translation and rotation information. On the other hand, far points provide accurate rotation information but weaker scale and translation information. We triangulate far points when they are supported by multiple views.

Monocular keypoints are defined by two coordinates on the left image and correspond to all those ORB for which a stereo match could not be found or that have an invalid depth value in the RGB-D case. These points are only triangulated from multiple views and do not provide scale information, but contribute to the rotation and translation estimation.

B. System Bootstrapping

One of the main benefits of using stereo or RGB-D cameras is that, by having depth information from just one frame, we do not need a specific structure from motion initialization as in the monocular case. At system start-up, we create a keyframe with the first frame, set its pose to the origin, and create an initial map from all stereo keypoints.

C. Bundle Adjustment with Monocular and Stereo Constraints

Our system performs bundle adjustment (BA) to optimize the camera pose in the tracking thread (motion-only BA), to optimize a local window of keyframes and points in the local mapping thread (local BA), and after a loop closure to optimize all keyframes and points (full BA). We use the Levenberg–Marquardt method implemented in g2o.

Motion-only BA optimizes the camera orientation and position minimizing the re-projection error between matched 3D points in world coordinates and keypoints, either monocular or stereo.

Full BA is the specific case of local BA, where all keyframes and points in the map are optimized, except the origin keyframe that is fixed to eliminate the gauge freedom.

D. Loop Closing and Full BA

Loop closing is performed in two steps, firstly a loop has to be detected and validated, and secondly the loop is corrected optimizing a pose-graph. In contrast to monocular ORB-SLAM, where scale drift may occur, the stereo/depth information makes scale observable and the geometric validation and pose-graph optimization no longer require dealing with scale drift and are based on rigid body transformations instead of similarities.

In ORB-SLAM2 we have incorporated a full BA optimization after the pose-graph to achieve the optimal solution. This optimization might be very costly and therefore we perform it in a separate thread, allowing the system to continue creating map and detecting loops. However, this brings the challenge of merging the bundle adjustment output with the current state of the map. If a new loop is detected while the optimization is running, we abort the optimization and proceed to close the loop, which will launch the full BA optimization again. When the full BA finishes, we need to merge the updated subset of keyframes and points optimized by the full BA, with the non-updated keyframes and points that where inserted while the optimization was running. This is done by propagating the correction of updated keyframes (i.e. the transformation from the non-optimized to the optimized pose) to non-updated keyframes through the spanning tree. Non-updated points are transformed according to the correction applied to their reference keyframe.

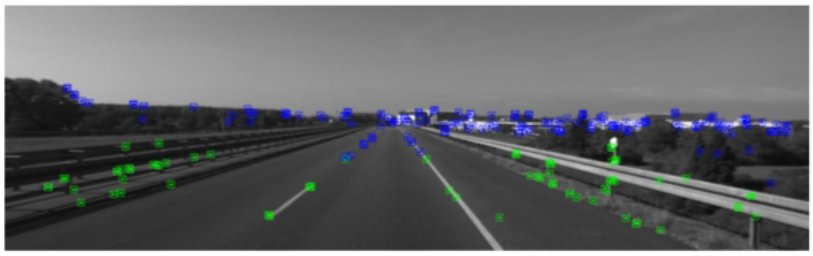


Fig. 3. Tracked points in KITTI 01 [2]. Green points have a depth less than 40 times the stereo baseline, while blue points are further away. In this kind of sequences it is important to insert keyframes often enough so that the amount of close points allows for accurate translation estimation. Far points contribute to estimate orientation but provide weak information for translation and scale.

E. Keyframe Insertion

ORB-SLAM2 follows the policy introduced in monocular ORB-SLAM of inserting keyframes very often and culling redundant ones afterwards. The distinction between close and far stereo points allows us to introduce a new condition for keyframe insertion, which can be critical in challenging environments where a big part of the scene is far from the stereo sensor, as shown in Fig. 3. In such environment, we need to have a sufficient amount of close points to accurately estimate translation, therefore if the number of tracked close points drops below a certain amount and the frame could create at least a specific amount of new close stereo points, the system will insert a new keyframe

F. Localization Mode

We incorporate a Localization Mode which can be useful for lightweight long-term localization in well mapped areas, as long as there are not significant changes in the environment. In this mode, the local mapping and loop closing threads are deactivated and the camera is continuously localized by the tracking using re-localization if needed. In this mode, the tracking leverages visual odometry matches and matches to map points. Visual odometry matches are matches between ORB in the current frame and 3D points created in the previous frame from the stereo/depth information. These matches make the localization robust to unmapped regions, but drift can be accumulated. Map point matches ensure drift-free localization to the existing map. This mode is demonstrated in the accompanying video.