Overviw by Jeff Lai

```
ORB_SLAM2 System SLAM //create the object of ORB_SLAM2 system
    mpVocabulary = new ORBVocabulary(); //get the bag of words of ORB
    //create the database of keyframe
    mpKeyFrameDatabase = new KeyFrameDatabase(*mpVocabulary)
    mpMap = new Map(); //create an object of map

    //create two displays
    mpFrameDrawer = new FrameDrawer(mpMap);
    mpMapDrawer = new MapDrawer(mpMap,strSettingsFile);

    //Initialize an object of tracking
    mpTracker=newTracking(this,mpVocabulary,mpFrameDrawer,mpMapDrawer,mpMap,mpKe
    yFrameDatabase,strSettingsFile,mSensor)

    // Initialize an object of Local Mapping
    myLocalMapper = new LocalMapping(mpMap,mSensor==MONOCULAR);

    //Initialize an object of Loop Closing
    myLoopCloser = new
LoopClosing(mpMap,myKeyFrameDatabase,mpVocabulary,mSensor!=MONOCULAR);

    //Display
    mpViewer = new Viewer(this,mpFrameDrawer,mpMapDrawer,mpTracker,strSettingsFile);
```

System:

1. Input image: $\begin{cases} GrabImageStero(imRectLeft, imRectRight) \\ GrabImageRGBD(imRectLeft, imRectRight) \\ GrabImageMonocular(im) \end{cases}$

2. Transform to grayscale: $\begin{cases} Stereo: mImGray, imGrayRight \\ RGBD: mImGray, imDepth \\ Mono: mImGray \end{cases}$

3.

Construct:

$$\begin{cases} Stereo: Frame(mImGray, imGrayRight, mpORBextractorLeft, mpORBextracotrRight) \\ RGBD: Frame(mImGray, imDepth, mpORBextractorLeft) \\ Mono(not\ initialize): Frame(mImGray, mpIniORBextractor) \\ Mono(initialized): Frame(mImGray, mpORBextractorLeft) \end{cases}$$

4. Tracking: flow into tracking thread.

Tracking:

1. Input: Frame

Initialization: $\begin{cases} \text{SteroInitialization()} \\ \text{MonocularInitializtion()} \end{cases}$

2. Estimation of camera pose

$$: \begin{cases} \text{mbOnlyTracking(False)}: \begin{cases} Estimation\ of\ pose \begin{cases} TrackWithMotionModel() \\ TrackReferenceKeyFrame() \end{cases} \\ Relocalization() \end{cases} \\ \text{mbOnlyTracking(True)}: \text{tracking and localization, no inesrtion of keyframe, no local map} \end{cases}$$

3. Track Local Map:

(1)UpdateLocalMap()$\begin{cases} UpdateLocalKeyFrames() \\ UpdateLocalPoints() \end{cases}$

(2)SearchLocalPoints():local map matches the current frame

(3)PoseOptimization():minimum the project error to optimize the pose

4. New KeyFrame Decision

More than 20 frames must have passed from the last global relocalization.

Local mapping is idle, or more than 20 frames have passed from the last keyframe insertion.

Current frame tracks at least 50 points.

Current frame tracks less than 90% points than Kref.

5. Construct keyframe: $\begin{cases} \text{KeyFrame(mCurrentFrame, mpMap, mpKeyFrameDB)} \\ \text{For RGBD, MONO: construct some MapPoints} \end{cases}$

**mbOnlyTracking is set as False at first.

Code Notes:

If the variable starts with ‘m’, it is a member variable.

For the first or second letter,

p	point
n	int
b	bool
s	set
v	vector
l	list
KF	KeyPoint

Core Functions:

TrackWithMotionModel(): tracking according to motion model, set the velocity and pose as the starting point, and projection optimization.

1. Predict the pose of current frame by the pose and velocity of last frame
2. Estimate the camera pose by PnP. The map points of the last frame are projected onto the frame plane of the current fixed size range. If there are fewer matching points, the range of picking points is doubled.
3. Run BA algorithm, Optimize camera pose by Least Square Method.
4. After optimizing the pose, the key points and map points of the current frame are discarded, and the remaining points are used for the next operation.

TrackReferenceKeyFrame():

Track according to the key frame, find the similar frame of Bow from the key frames, and optimize the pose by matching.

1. Solve the BOW vector of the current frame.
2. Search for key point matching between the current frame and the key frame. If the matching relationship is less than 15 pairs, Track fails.
3. Set the location of current frame as that of last frame.
4. Discard useless points. If the number of matching is more than 10 pairs, return true.

Relocalization():

Find out the candidate frames with sufficient matching points between the previous key frame and the current frame, use Ransac iteration and solve the pose through PnP.

1. Firstly, the BOW value of the current frame is calculated and the candidate matching key frame is found from the key frame database.
2. Construct PnP solver, mark clutter, prepare matching point set for each key frame and current frame
3. PnP algorithm is used to solve pose, several P4P Ransac iterations are carried out, and non-linear least squares optimization is used until a camera position with sufficient inliers support is found.
4. Return True or False

MonocularInitialization():

Accurate initial values are obtained by pole constraints and global BA optimization between the first two frames.

1. When first run, there is no previous frame data, the current frame is saved as the initial frame and the last frame, and an initializer is initialized.
2. Run secondly, it has an initializer now.
3. The ORB matcher is used to match the current frame and the initial frame. When the corresponding relationship is less than 100, it fails.
4. The homography matrix and the basic matrix are computed by two threads with the polar constraints of the eight-point method, and the score is used to determine whether the homography matrix is used to restore the trajectory of motion or the basic matrix is used to restore the trajectory of motion.
5. Set the initial frame and current frame as key frame, build MapPoint
6. Optimize camera pose and coordinates of key point by BA
7. Set unit depth and zoom in the initial baseline and map location.

TrackLocalMap():
Through projection, find more corresponding relations and accurate results from the generated map points.
1. Update Covisibility Graph, update local key frames.
2. Update the local map points according to the local key frames, and then run the filtering function isInFrustum.
3. Projection of map points onto the current frame, discarding beyond the image range.
4. Current line-of-sight direction V and average line-of-sight direction n of map point clouds, discarding point clouds with n*v<cos(60)
5. Abandon points whose distance from map point to camera center is not within a certain threshold
6. The end of isInFrustum function for calculating the scale factor of an image
7. Nonlinear least squares optimization
8. Statistical updates of map points.