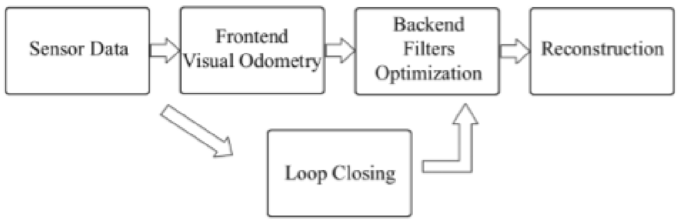
This is my understanding about a slam system after learning slam

A classic slam frame consists of 5 parts



Among them, frontend, backend, and loop closing are all focused on Localization, and reconstruction is focused on Mapping;

Common sensors include Camera(mono camera/stereo camera, RGB-D), IMU, Radar and Lidar.

Frontend Visual Odometry(VO):
Estimate the movement between two adjacent frames, and triangulate the spatial position of landmarks;

Backend Optimization:
Accept the camera poses and landmarks solved by frontend and optimize ;

Loop Closure Detection:
Determine whether the robot has reached the previous position

Mapping:
According to the estimated trajectory, create a map with the mission requirements

Frontend methods and processes:

->Define a Mat, read the incoming image and initialize it, if it is stereo or RGB-D, it will solve some 3D-landmarks during initialization to set an initial Map.

-> Extract the feature points of the current frame image, to match/track with the previous frame. There are three methods to achieve
1. Feature method:
extract descriptors and brief to match;

2. Optical Flow (for example LK):
Extract corner points, solve the least square problem, which based on the assumption of gray invariance, to find the pixel block, which is closest to the corner point in gray in current frame , and obtained result is the pixel coordinate in current frame;

3.Direct Method:

Extract corner points, according to the assumption of gray invariance and triangulation of landmarks, by projecting each landmark can get two pixels coordinaten, in current frame and last frame, calculate their luminosity error/brightness error, there should be many landmarks, therefore, an optimization problem can be constructed and the least squares solution can be obtained. The result of the optimization is the **pose** of the current frame to the previous frame;

->If Frontend use the Feature Method or Optical Flow to match the feature points of the two frames, to get the pose of frames, it will be necessary to continue to solve the relative motion between the two frames with these matching features.
That is why Direct Method is more quickly and can calculate more features.

There are two ways to solve the relative motion

1.2D-2D:
Epipolar geometry, 8-point method

2.3D-2D PnP (perspective-n-point):
The premise of this solution is to have some coordinate points (3D-landmarks) in three-dimensional space, which have matching feature points on some frames. The feature points of last frame are tracked/matched, the corresponding landmark coordinates are inherited/obtained by features in current frame, and the landmarks are reprojected (T*P, this T is the pose to be optimized, and the pose of the previous frame can be used as the initial value) to the current frame as a new coordinate (2D-pixel), there will be an error between these reprojected 2D-pixel coordinates and the coordinate of the corresponding feature point, so a set of equations constructs an optimization problem, and the pose will be solved;

-> There should also be a mechanism to eliminate outliers in the front end. For example, when matching or tracking feature points, it is normal for mismatches to occur; during optimization, if a certain error is found to be particularly large, it means that the feature points are probably not correct matched, it will affect the result of optimization, so it should not be introduced in the next optimization.

-> In particular, if there are fewer points tracked, it means that the pose of the current frame has changed significantly in compare with tracked frame, then the current frame is judged to be a new key frame, and the following processing is performed:

1. Extract new feature points
2. Find the corresponding points of features on the right (if use sterero camera), then triangulate to create new landmarks
3. Insert new keyframes and landmarks into the map, and trigger a backend optimization

-----------Frontend for this Frame is finished, wait for next frame------------

Backend Optimization:

-> The front end solves the coordinates of the landmarks, and the poses of different frames.

First of all, in order to meet real-time performance, the results of the Frontend solution are a bit rough, and they are only short-term trajectories and maps. Due to the accumulation of errors(Drift accumulate), they are inaccurate, so it is necessary to further construct a larger scale and scale optimization to make the long-term motion trajectory and map mapping accurate;

Common optimization methods include Kalman filter and nonlinear optimization. In most mainstream open source slam frameworks, such as ORB-SLAM or VINS-Mono, Backend optimization is nonlinear optimization. There are reasons for this:

1. The filter method often makes the Markov assumption, that is, the state at time k is only related to time k-1, and has nothing to do with the time before k-1. In visual odometry, it means that the current frame is only related to the previous frame. It has nothing to do with other frames. This assumption is sometimes not so reasonable in slam. For example, in loop detection, the current frame needs to be associated with the frame long ago.
Non-linear optimization not only considers landmarks and poses at neighboring moments, but also includes more previous historical data. It uses more information, and of course, requires more calculations.

2. Filtering methods such as EKF need to store the mean and variance of the state quantity, as well as the covariance matrix. If you consider the landmark, because there will be a large number of landmarks in the slam, the required storage capacity is too large and it is not suitable for large-scale scenes.

3. if it is EKF, only one linearization is performed to solve the posterior at X(k-1). When the nonlinearity of the slam motion equation and the observation equation are both strong, the linear approximation can only be established in a small range, if the working point is far away, it will be not accurate;

Non-linear optimization GN or LM is a second-order approximation, and during every iteration, it will re-expand near the new estimated point, which has a larger application range, instead of just doing a Taylor expansion at a fixed point like EKF;

4.There is no anomaly detection mechanism in filter methods such as EKF (threshold can be set to eliminate outlier in the process of nonlinear optimization), which will cause the system to diverge when there are outliers. However, in slam, outliers are common and if there is no anomaly detection mechanism, will make the system unstable in practice.

Therefore, it is generally believed that the nonlinear optimization method has better results under the same level of calculation.

The mainstream slam framework in recent years generally chooses the nonlinear optimization method in Backend

There are two common optimization algorithms: Gauss-Newton, Levenberg-Marquardt
No matter which method you choose, you need to provide an initial value of the variable when optimizing.

This initial value is often given by the front end by solving the epipolar geometry/ICP/PnP

Taking the Gauss-Newton method as an example, there are generally three implementation methods in code:

1. using Eigen library write Model
2. Using ceres library to build an optimized solution model, configure the solver, and automatically find the derivative
3. Using g2o library to define graph optimization model and configure optimizer

Comparing the calculation time, in general, using Eigen to write Model directly is the fastest, followed by g2o library, and finally ceres library. Because for both Eigen and g2o we need to write jacobian by ourselves, but ceres does not need it. It supports automatic solve Derivative, so it seems more time-consuming. This also satisfies our understanding, for a method, the better the versatility, the worse the efficiency.

Of course, because the amount of calculation for the Backend Optimization is greater than that of the Frontend, and the computing power of the equipment is limited, so it is impossible to optimize all camera poses and landmarks without limitation (if we call it a Bundle Adjustment Problem).
The scale of BA needs to be controlled. Solution is to extract Key-Frames and set sliding window.

1. Extract key frames

The Backend does not need to deal with all the frames as Frontend. For example, in VINS-Mono, the camera can release 30 Frame/s to the front end, it means, the time interval between frames is about 0.033s, in this case Even if driving at a high speed of 30m/s, no matter driving in a straight line or tsteering, the displacement between the two frames is less than 1 meter, and the steering angle is also small,

so it can be considered that two images obtained between the current frame and the previous frame in content is very similar. The effect of BA for these two frames and the effect of BA for only one of the frames don't show much difference.

Therefore, it is not necessary to optimize each frame. For example, for the successive frame, we can try to extract a key frame at a three-frame intervals , or if the pose of the current frame has changed significantly comparing with last keyframe, send it to the back-end for optimization,;

2. Sliding window

Although there are key frames to reduce the number of processing, as time accumulates, the scale of the map continues to grow, and the number of key frames will increase. In order to control the BA scale, choose to keep only the N key frames closest to the current frame and remove the time Earlier keyframes.

Order sequence of backend optimization:

->Through multi-threading tool, after the front end completes the calculation of the new key frame and landmark points, wake up the back end thread

-> The back-end thread optimizes the objects in the given key frame and landmark point list. It only sets the frames and landmarks in the sliding window, according to the criterion of minimizing the reprojection error ->Update the pose and landmark after the optimization is completed, the positioning is completed, and then the mapping is triggered

Build a map

The map can be considered as a collection of all landmarks. According to different mapping requirements, the types of maps obtained are also different.

1. Positioning -> sparse road sign mapping
2. Navigate / avoid obstacles / reconstruction -> dense map
3. Interactive/AR->Semantic Map

If you want to build a dense map for navigation, using monocular and binocular cameras based on stereo vision to achieve it is not thankful, the implementation is complicated, the calculation is large, and the depth estimation is unreliable, especially monocular Is fragile;
The effect of dense reconstruction using RGB-D is better. It can directly measure the depth without consuming a lot of computing resources to estimate, but it has a narrow measurement range, small field of view, large noise, and easy to be interfered by sunlight. It is generally used indoors and outdoors. Difficult to adapt

Therefore, if we want to make dense mapping in large scenes, it is best to use Lidar to do a multi-sensor fusion. representative work have LOAM, LIO, LINS.