Paper-choosing: MonoSLAM: Real-Time Single Camera SLAM

a) During the previous years, many teams had kept working on SLAM and got their processing, but still have some unsolved problems, but which means we can do further research and practical work base on them. For example, Harris and Pike had achieved 3D maps of long image sequences features a later real-time implementation, but ignored the strong relationship between the estimation of mapped vision features and the camera motion. And this problem was also ignored by another team, which indicates it is hard to work out it. Also, EKF was used to update localization and mapping problems, the deviated EKF is mainly for the construction of large-scale maps. Moreover, for Vision-based SLAM, Neira and his/her team worked a base on the paper written by Davison and Murray. They made the system create 3d natural landmarks and a robot that can choose features to estimate and do the sparse mapping. Another team worked on a stereo vision SLAM, which has sequential implementation but not real-time, some other teams combined omnidirectional vision with other sensors, and their solutions can be well extended to large-scale mapping when using effective sparse information filters if there are fewer loop closures. Since some algorithms are really expensive and high computational cost, one of the authors' team's works was to simplify the hardware requirements of SLAM.

   And also, generally, to figure out the problems mentioned before, the team has worked out the real-time algorithm for the 3D trajectory recovering of a fast-moving monocular camera under an unknown environment. They were focused on high frame-rate real-time performance and chose localization as the main point. Especially the repeatable localization, which means gradual drift from ground truth does not occur, is the most important case.

b,c) Firstly, the algorithm is the first successful application of the SLAM methodology from mobile robotics to the "pure vision" domain of a single uncontrolled camera, achieving real-time but drift-free performance inaccessible to Structure from Motion approaches. The significance of this contribution is they updated the SFM algorithm from offline to online, when the robot is moving, it can analyze and create a sparse but persistent map of the image observation and camera trajectory.

   Secondly, they had an active approach for mapping and measuring, using a general motion model to make the camera move smoothly, also figuring out the monocular feature initialization and feature orientation estimation. For the significance of this contribution, this had made them reach their expect: the high frame-rate real-time performance, which I mentioned in answer for (a), and the frequency is typically 30 Hz(with standard PCs and cameras). And all of these formed a very solid algorithm. Moreover, this work has extended the user of SLAM in the robotic region.

   Thirdly, the key contribution is they proved that for a fasting-moving robot with a camera on it, and that is the only data source when the robot moves in a random region, the algorithm can still achieve real-time localization and build a not only detailed but also persistent map. For the significance of this contribution, I think is they used a feature of SLAM, which can deal with the image processing, tiny search regions and data, which they really needed, ignore all the useless data, this made the work more efficiently in total, and found a new application for SLAM.

   To build the probabilistic 3D Map, they used a state vector x^ and covariance matrix P to represent a map, and feature states yi are the 3D position vectors of the locations of point

features. The main proposal of this map is to permit real-time localization and then capture the sparse set of high-quality landmarks. The SLAM here has the complexity O(N^2)(N: number of features), which bounds the N for real-time processing around 100. And use the single, full covariance EKF to SLAM to have a long-term, repeatable localization in a limited scope.

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \vdots \end{pmatrix}, \; \mathrm{P} = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} & \cdots \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} & \cdots \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}. \qquad \mathbf{x}_v = \begin{pmatrix} \mathbf{r}^W \\ \mathbf{q}^{WR} \\ \mathbf{v}^W \\ \omega^R \end{pmatrix}.$$

To find the natural visual landmarks, improve matching over large camera displacements and rotations will extend the features' powers. Use image patches as landmark features to improve matching performance based on camera location information. Use Shi and Tomasi operator to detect corner points, here is a grayscale image(more efficiency). For matching, it is assumed that the pixels of the image block of each landmark point is coplanar(this assumption can work well in most cases).

After choosing some known targets placed in front of the camera for initializing to finish system initialization, motion modeling and prediction have to be done. Since there was no such prior information for the agile camera's movement, any model will stop at a certain level of detail, using probability assumptions to represent the difference between the built model and the real model, so use the constant velocity, constant angular velocity model is better. Suppose for each time step, and it has unknown acceleration a^W and angular acceleration alpha^W, which obeys 0-mean Gaussian distribution and has an impact on velocity and angular velocity:

$$\mathbf{n} = \begin{pmatrix} \mathbf{V}^W \\ \Omega^R \end{pmatrix} = \begin{pmatrix} \mathbf{a}^W \Delta t \\ \alpha^R \Delta t \end{pmatrix}.$$

Although V^W and Omega^R may be coupled, now can be assumed that the covariance matrix of n is a diagonal matrix, which means that they are not related, the updated state:

$$\mathbf{f}_v = \begin{pmatrix} \mathbf{r}^W_{new} \\ \mathbf{q}^{WR}_{new} \\ \mathbf{v}^W_{new} \\ \omega^R_{new} \end{pmatrix} = \begin{pmatrix} \mathbf{r}^W + (\mathbf{v}^W + \mathbf{V}^W)\Delta t \\ \mathbf{q}^{WR} \times \mathbf{q}((\omega^R + \Omega^R)\Delta t) \\ \mathbf{v}^W + \mathbf{V}^W \\ \omega^R + \Omega^R \end{pmatrix}.$$

A quaternion represents the attitude, and the quaternion is calculated by the angular axis (omega^R+Omega^R)Delta t. In EKF, the new state estimation f_v(x_v,u) obtained by the camera through the motion equation must be accompanied by the increase of the state uncertainty (process noise covariance) Q_v. Q_v is calculated by Jacobian:

$$\mathrm{Q}_v = \frac{\partial \mathbf{f}_v}{\partial \mathbf{n}} \mathrm{P}_n \frac{\partial \mathbf{f}_v}{\partial \mathbf{n}}^\top,$$

P_n is the covariance matrix of the noise vector. The uncertainty growth rate in the motion equation is determined by P_n, and its magnitude represents the smoothness of the motion that is expected.

To predict the image position before the observation is the main point for active feature measurement and map updating. Feature matching uses NCC for template matching. First, use the state of the camera pose to estimate x_v and the feature position y_i, so the position of a point feature relative to the camera is expected to be:

$$\mathbf{h}^R_L = \mathrm{R}^{RW}(\mathbf{y}^W_i - \mathbf{r}^W).$$

Using the standard pinhole model, the location where the feature is expected to be found in the image (u,v):

$$\mathbf{h}_i = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 - fk_u \frac{h^R_{Lx}}{h^R_{Lz}} \\ v_0 - fk_v \frac{h^R_{Ly}}{h^R_{Lz}} \end{pmatrix},$$

(where fku, fkv, u0, and v0 are the standard camera calibration parameters)
And then use radial distortion to distort the perspective-projected coordinates u=(u,v) to obtain the final predicted image position u_d=(u_d,v_d). The reversible radial distortion model used is as follows:

$$u_d - u_0 = \frac{u - u_0}{\sqrt{1 + 2K_1 r^2}},$$

$$v_d - v_0 = \frac{v - v_0}{\sqrt{1 + 2K_1 r^2}},$$

Calculate the Jacobian matrix of the two-step projection function to the camera and the feature separately, which can calculate the uncertainty of the position of the predicted feature on the image, represented as the 2\times2 symmetric covariance matrix S_i:

$$\begin{aligned} \mathbf{S}_i = & \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v} \mathbf{P}_{xx} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v}^\top + \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v} \mathbf{P}_{xy_i} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{y}_i}^\top + \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{y}_i} \mathbf{P}_{y_i x} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v}^\top \\ & + \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{y}_i} \mathbf{P}_{y_i y_i} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{y}_i}^\top + \mathbf{R}. \end{aligned}$$

Avoid tracking the new features of several frames in the image without trying to estimate their 3D position, and then use multi-view geometric triangulation to perform minibatch to estimate the depth of the initial features. Their approach was to initialize a 3D ray in the map, the starting point is the location of the camera, pointing to the feature point.
Finding features used Shi-Tomasi operator, and image patch is used for matching. Moreover, the search box is randomly selected, then the feature points are found in it. The requirements of the search box selection cannot overlap with the existing features (including the newly extracted current frame). The constant velocity model here is used to predict the possible occurrence. The locations of these predictions are not allowed to extract feature points.
When initializing the feature, it is not the feature descriptor but the image patch used for feature matching. Once the feature is initialized, the image patch will no longer change. In order to solve the problem of rotation invariance, the paper uses the before matching predict how the image might be seen first to correct the viewing angle problem caused by camera movement to a certain extent. The transformation equation for feature direction estimation is:

$$\mathbf{H} = \mathbf{C}\mathbf{R}[\mathbf{n}^T \mathbf{x}_p \mathbf{I} - \mathbf{t}\mathbf{n}^T]\mathbf{C}^{-1},$$

(C is the camera correction matrix (perspective geometric projection), R And t is the pose of the camera, n is the normal vector, and x_p is the projection of the center of the image patch on the image). After a more efficient step, globally is found to be the best fit. To solve the uncertainty of adding new features to the map, the normal estimates are in a separate two-parameter EKF for each feature.
For the experimental part, they used a cheap IEEE 1394 web hand-held camera move around a room to show how to stably add virtual furniture to a 30Hz image stream. Then the MonoSlam was able to estimate the motion of the hand-held camera in a real-time image stream and provide it directly to the rendering engine in Augmented Reality(AR). Also, in

implementation, the linear acceleration noise components in P_n set to a standard deviation of 10ms^-2 (1 acceleration due to gravity), and angular components with a standard deviation of 6rads^-2. The magnitude of these accelerations empirically describes the approximate dynamics of the camera moving quickly and smoothly in hand (except very sudden and violent movements). When the camera continued to move, these objects would be fixed in the scene in the image view. Users can be presented with displays of features highlighted image stream and the estimated position of the camera.

For a humanoid robot platform HRP-2, it also presented with MonoSLAM, so there was only a sparse map of points landmark. Although some high-acceleration 3D motion happened, the algorithm still formed a persistent map that with drift-free real-time localization on a small area. HRP-2 can perform accurate 3D measurement in the focused observation area in front of the robot with a high-performance forward-looking trinocular camera rig, and equip with a 3-axis gyro in the chest for reporting measurements of its angular. Set the gyro to 30 Hz and evaluated each element's standard deviation of the angular velocity measurement as 0:01rads^-1, which is an additional Kalman update step before visual processing. They also gave a truly SLAM experiment: let the robot walk in a circle with a radius of 0.75m. The result: classic SLAM behavior was observed, and the uncertainty of the new mapping features steadily grew until the early features were reobserved, the loop closed, and the drift was corrected. Furthermore, found a map of features that can fit long-term use, which can complete any number of loops in positioning accuracy without drifting.

d) Since the MonoSLAM algorithm has achieved some expect, but still have some limitations. Firstly, the cameras were all set in an indoor place with not too much decorations to detect features of objects then to do further work, so it is hard for the algorithm to solve some complex scenes with too much stuffs, both indoor and outdoor.

Secondly, for the more dynamic motions, the MonoSLAM may not work efficiently like before due to there was only one free-moving camera, the camera may miss some actions.

Thirdly, suppose it is used in a more complex outdoor scene, then it is not sufficient for things with lots of details. And there may be sun, so the lighting and shade are keep changing. These will also influence the results.

Fourth, it is clear that orientation estimation only works well with large patches and have significant interesting texture over their area.

The next possible step is focusing on hard-real operation, commodity cameras, and minimal assumptions. To do this, one possibility is to explore cameras that can capture when the rate is more than 30Hz, the highly increase of frame rate will not make the work on image processing increase in the same way. And due to the decreasing motion uncertainty, the search regions will become smaller as well. For using MonoSLAM in a larger and more complex scene, the submapping strategy maybe work. However, it is also a problem to have meaningful divided subblocks. Once done that, it will be able match submaps first, then join the small-scale mapping network through a relatively loose set of estimated transformations. And last, small and large loops are routinely and seamlessly closed by the system. An active feature selection mechanism can lead to a good result.