Monocular Visual Odometry for Indoor Environment with Initial Map Building with LIDAR Input

Sha Yi, Chaohui Gong, Howie Choset

***Abstract*—Visual Odometry is widely used for mobile robot localization because of its relatively low cost compared with high precision sensor system, such as LIDAR. However, pure visual odometry is usually of low accuracy, which is caused by accumulated drift, inaccurate feature matching, inaccurate scaling factor, and unstable environment. In this work, we propose a system of combining the pose from LOAM (LIDAR odometry and mapping) and visual feature points (ORB) to build an initial map of the environment, then localize a mobile robot with only a monocular camera onboard and build a denser map with reference to the initial map to enhance accuracy compared with pure monocular visual odometry. In this way, the localization accuracy of a mobile robot in indoor environment could be significantly improved with a single LIDAR, of which the one-time cost is ignorable for a large amount of robots and for a long term.**

***Index Terms*—Monocular Vision, Localization, Visual Odometry.**

I. Introduction

Pose estimation is one of the most vital parts in robotics. Some of the key tasks for robots: path planning, object avoidance etc. all rely on the localizing the robot and do computation based on it.

There are multiple ways to do localization. Lidar has become popular these years for localization. It makes use of laser to obtain point cloud of the environment and compare adjacent point cloud to localize the robot. It is general of high accuracy, but the cost for Lidar system is much higher than most of the visual localization system so that industrial applications tend not to use Lidar on every robot. Some low cost solutions are built with visual systems. For example, Apriltag [5] is a popular approach to get the pose of camera. They are special features encoded for pose detection. Apriltag approach is relatively cheap [5], fast, and very accurate in short distance. However, it is difficult to maintain tags in industrial environment and the accuracy is not high for long distance [5]. Visual Odometry is another visual localization solution that reduces cost. However, it is not as accurate as the previous solutions since it accumulates drift during the process of localization.

Visual odometry localizes the robot from Structure from Motion [1]. The process of doing structure from motion is:

- Read the image frame from the camera sensor
- Detect visual feature points from the image frame
- Extract feature descriptors from the visual feature points
- Search between the features points of consecutive frames for feature matching
- Calculate Essential Matrix/Fundamental Matrix
- Get pose (rotation and translation matrix) from the Essential Matrix/Fundamental Matrix

We may additionally do triangulation to get the 3D location of the matched feature points, so that it is possible to use those 3D points for mapping. However, the pure visual odometry is not of high accuracy.

In this work, we propose an indoor mapping and localization system for factory or office scenario, which consists of one-time mapping and long-term visual localization. To build an accurate visual feature map of the environment, a mobile robot equipped with Lidar and cameras will explore the environment and log the pose from LOAM [8] and corresponding visual feature points in camera image. The map is stored for further visual localization untill the environment has significant change, when a re-mapping is required. With the visual feature map, a mobile robot with only a low cost camera on board can accurately localize itself with no accumulated drift effect.. In this case, several companies may share the cost of one Lidar system and they may only need it again when the company change their environment. Therefore, the cost of localization could be greatly reduced. The system operates in the following process:

- Do the first four steps as in the previous structure from motion procedure
- Read the Lidar pose
- Do triangulation on the matched feature points based on the input Lidar pose
- Get the 3D location of the visual feature points

With those feature points, it is possible to build and save a 3D map of the environment and the robot could localize itself later in the tracking process.

In this work, we used the commercialized LIDAR system Stencil from Real Earth [8] for pose estimation, and ROS (Robot Operating System) [6] for hardware interface and communication.

S. Yi is with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic Unverisity, Hong Kong email: 12132054d@connect.polyu.hk

C. Gong and H. Choset are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA, 15213, USA. emails: chaohuig@andrew.cmu.edu, choset@cs.cmu.edu

II. Related Work

A. Project Tango

Google developed the Project Tango [2] system to do visual odometry for localization. The Tango system fuses the visual estimation with the pose information of inertial measurement unit (IMU). The system operates by creating an initial map of the environment. However, the Tango system is not open source, thus there is little flexibility to build and modify with industrial specifications. Our goal is to develop a system to replace the dependency on Project Tango on the current robot.

B. ORB-SLAM

Mur-Artal et al. [3] proposed the ORB-SLAM system that can be used for monocular vision. It is a pure vision approach so that it is impossible to get the scaling factor of the odometry. The system structure is shown in Fig. 1:

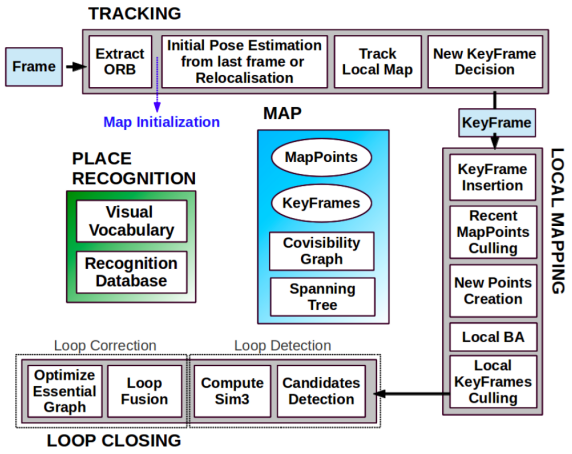

Fig. 1: ORB-SLAM System [3]

The ORB-SLAM system combines most of the popular visual odometry approaches including bundle adjustment [7], loop closing, key frame culling, etc. The monocular system runs with an initialization process with structure from motion first, then assumes a constant velocity motion model and only do least square optimization on this model. Therefore, it is relatively robust and fast enough. Thus, we build our system on the ORB-SLAM and since it is open source, we do our modification on the code it provides.

III. Methodology

A. Triangulation

The 3D location of feature points are computed using triangulation [1]. We could obtain the previous pose $g_{l0} = \{ R_{l0} \mid t_{l0} \}$ and the current pose $g_{l1} = \{ R_{l1} \mid t_{l1} \}$ from LIDAR odometry. We can then get the camera pose:

$$g_c = g_l^c g_l \tag{1}$$

where g_l^c is the transform from LIDAR to camera. Therefore the Projection matrix could be obtained:

$$\boldsymbol{P_c} = \boldsymbol{K}\,[\boldsymbol{R_c}|\boldsymbol{t_c}] \tag{2}$$

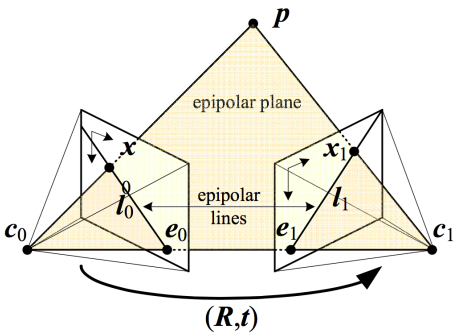

Fig. 2: Epipolar Geometry [1]

where $\boldsymbol{K}$ is the 3-by-3 intrinsic matrix of the camera.

As shown in the Fig. 2, $\boldsymbol{c_0}$ and $\boldsymbol{c_1}$ are camera centers which could be obtained from

$$\boldsymbol{t_j} = -\boldsymbol{R_j c_j} \tag{3}$$

Then the direction of rays $\hat{\boldsymbol{v}}_{\boldsymbol{j}}$ as shown in the Fig. 2 could be obtained [1]:

$$\hat{\boldsymbol{v}}_{\boldsymbol{j}} = \boldsymbol{R_j^{-1} K_j^{-1} x_j} \tag{4}$$

where $\boldsymbol{x_j}$ is the homogenous coordinate of each 2D feature point. Then the optimal point $\boldsymbol{p}$, as proven in [1], could be computed with:

$$\boldsymbol{p} = \left[\sum_j (\boldsymbol{I} - \hat{\boldsymbol{v}}_j \hat{\boldsymbol{v}}_j^T)\right]^{-1} \left[\sum_j (\boldsymbol{I} - \hat{\boldsymbol{v}}_j \hat{\boldsymbol{v}}_j^T)\boldsymbol{c}_j\right] \tag{5}$$

Then all the triangulated feature points will be saved as the map of the environment.

B. Map Building

With the pose input from LIDAR [8], it is not necessary to do the initialization process for the monocular tracking. In this case, we replace the initialization process with map building. The detailed process is shown in Fig. 3.

The reprojection process is done by triangulation mentioned in the previous section.

Then Least-square method Levenberg-Marquardt algorithm [7] is used to optimize the reconstructed map points, with pose fixed. The reason for this is that the pose output from Stencil LIDAR system is accurate enough, within 2 cm, to act as ground truth for the measurement.

The loop closing thread will keep detecting loop throughout the map building process. When a loop is detected, global bundle adjustment will start to avoid the drift generated from the visual odometry and optimize the 3D map points once again.

Thoughout the entire map building process, all the original key frame culling algorithm and reference frame tracking is kept. Therefore, it is more convenient for later tracking process since it preserves optimal information of the initial environment and odometry information.

When all the above process is done correctly, the map, map points, essential graph, key frames, key frame database are saved to the file system for future use.

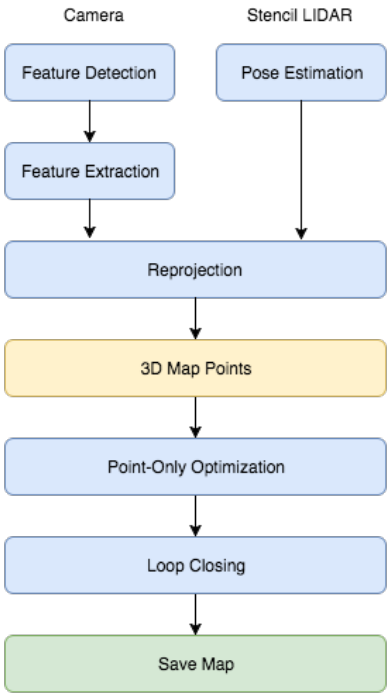

Fig. 3: Map Building Process

C. Tracking

During the tracking process, most of the original ORB-SLAM is kept for the system. The major difference is that the new system has a comprehensive ROS wrapper that broadcast topics on its current pose, initialization stage, number of features points so that it is easier to record data.

There are still three threads, tracking, loop closing and local mapping. The local mapping thread will be called once 3 seconds to relocate the robot with reference to the initial map built in the previous process.

D. Bag of Words

Nister and Stewenius [4] suggest the use of indexing feature descriptors to build a vocabulary tree for scalable recognition. In our case, vocabulary tree is built before hand and we make use of this for search on feature matching. As the ORB-SLAM [3] system suggested, the vocabulary tree is built on the detection of comprehensive dataset of indoor office environment, thus it fits our need in this case.

The bag of words matching will assign different clusters to feature point descriptor and only those that lie in the same cluster will be matched. This improves the searching speed to a great extent compared with brute force searching and matching of all feature points.

E. Constant Velocity Model

To speed up tracking process, the system is assuming a constant velocity model [4]. It makes assumption that when the camera frame rate is fast enough, the speed is more or less constant between adjacent frames.

Then when a new frame comes into the system, it first predicts its camera pose by multiplying constant velocity to the previous frame pose. The previous velocity v between pose g_0 and g_1 could be represented as:

$$v = g_0^{-1} g_1 \tag{6}$$

Therefore, by assuming constant velocity, the next pose g_2 could be predicted:

$$g_2 = g_1 v \tag{7}$$

Then least-square method is made on the current frame pose g_2. It takes the visible 3D map points and projects onto the current frame and minimize the projection error on the frame.

IV. Tests and Results

All the tests are done with the default ORB-SLAM parameters [3]. LIDAR is running with the Localization mode [8]. The robot we are using is shown below in Fig. 4:

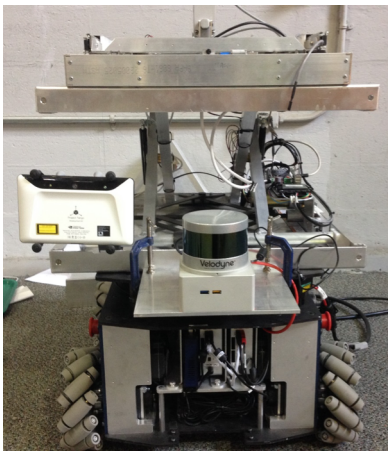

Fig. 4: Intelligent Mobile Robot (IMR)

A. Constant Velocity Model

The result is shown in Fig. 5 with a 10 Hz monocular camera, taking LIDAR [8] as ground truth.

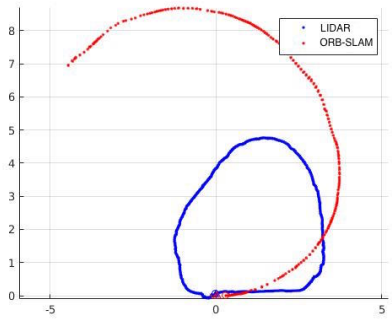

Fig. 5: Constant Velocity Model Test Result

However, as seen in the graph, speed is not always constant and it performs extremely bad when doing sharp turns. Though it might be better with a high frame rate camera, it is still necessary to have a better motion model to speed up the matching and optimizing process.

B. Joy Stick Twist Input

Since the constant velocity model is not accurate, we tried to use joy stick twist command information as motion model for prediction and do optimization on the predicted pose, i.e. replace the velocity v from the previous pose with the velocity input from the joy stick twist command. The result is in Fig. 6.

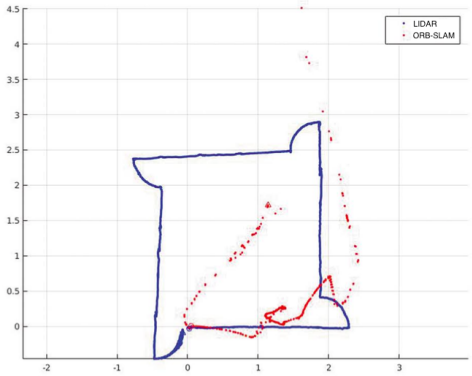

Fig. 6: Joy Stick Input Test Result

However, the joy stick twist information is far more inaccurate than the desired result.

V. Discussion

A. Image Processing

During the tracking process, it is very easy to lose track, especially when encountering challenging environment. These challenging environments include feature-less environment like walls and floor, and scenes that only have high contract and simple pattern.

To enhance the robustness, image filter is added to the camera frame before inputting to the visual odometry system. The current filter is a sharpen filter that deducts the original image with a Gaussian blurred image and this reduces the number of track lost on the snake robot when watching the floor.

B. Data Security

Another concern of using initial map instead of mapping every time when localizing is for the security reason.

LIDAR operates by collecting and comparing dense point cloud for mapping and localization, which means the detailed shape of surrounding object could be reconstructed from the LIDAR point cloud. It is possible that some of the factories or industries are high concerned about privacy so that they do not want their entire environment to be reconstructed by LIDAR point cloud so that people might hack into their system and get all the details of their environment.

The map-based visual odometry only store map point and is mapped offline every time after building the map. This is a much more secure and stable way if the environment does not change regularly.

VI. Future Work

A. Multiple Camera Pose Fusion

During our tests, we found that cameras on different position have various accuracy for the movement. For instance, when doing a left turn, the camera on the right side gives more accurate result than the left one, since it has longer trajectory and the features points move more obviously than those on the left. In another case when moving straight forward, the front camera might not provide as good pose result as the side cameras, since pin hole camera is better for rotation than translation or scaling.

If we could fuse multiple odometries together dynamically by taking the body twist as reference, we may obtain much better result than the result from only one camera.

B. Shaky Vision

In some cases, when the motion is very noisy, like on snake robot, the system is more likely to lose track because of blurred image or the drawback of constant velocity model.

One way to enhance the robustness during shaky motions is to change the camera with a better camera that has higher frame rate. Another way is to replace the constant velocity model with other models. For instance, if the snake robot is moving in a circular motion, we may input the circular motion as a reference so that it might make optimization easier.

C. Information Gain Evaluation

Since the visual odometry is usually connected with path planning or related decisions, it is possible to evaluate the environment and do better planning on this. For example, when the robot is about to turn to a clean wall or feature-less ground, by calculating the information gain we might be able to tell the upcoming scene is good or bad for the visual system. If the result is bad, we might need to avoid those scenes so that we get a more robust automation system.

VII. Acknowledgement

Acknowledgement to Chaohui Gong for guiding me all the way. Thanks to Elena Morara, Alexander Ansari, Lu Li, Zhongqiang Ren, Jin Dai, Puru Rastogi for giving suggestions. Thanks to Prof. Howie Choset for having me in the Biorobotics lab.

References

[1] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[2] R. Dugan J.C. Lee. Google project tango.

[3] Raul Mur-Artal, JMM Montiel, and Juan D Tardós. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[4] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. IEEE, 2006.

[5] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3400–3407. IEEE, 2011.

[6] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[7] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[8] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems Conference (RSS)*, pages 109–111, 2014.