Vision and LIDAR Feature Extraction

Yong-Way Chee (yc563), Tsung-Lin Yang (ty244), *Cornell University CS4758 Robot Learning*

***Abstract*— In this project we will discuss about methods of extracting stable features for camera and laser range finder in order to use these features in SLAM. The methods including dot-product feature extractor and line segment feature extractor for laser range finder, corner feature extractor for camera images. The robot platform we work on is the Segway RMP 50.**

I. Introduction

When a robot is placed in an unknown environment, it is essential for the robot to know where it is located relative to its environment in order to perform tasks such as search and rescue. The technique often used is called Simultaneous Localization and Mapping, or SLAM. In order to accomplish SLAM, feature extraction is a critical step which will direct affect the performance of the SLAM.

The FAST corner detector algorithm presented in Rosten et al. [4] is a high performance algorithm for feature extraction on images. OpenCV is an open source library which has useful functions for solving various computer vision problems. Particularly, the GoodFeaturetoTrack function will analyze an image to find distinct features in order to perform other tracking algorithm. It also has optical flow function which tracks a pixel's relative movement between two different frames by processing a source pixel's intensity and relative position. Similarly, laser range data can also be analyzed to find features. Line features or corner features are possible candidates of good stable features for SLAM. The objective of this project is to find and implement a good algorithm to extract features for SLAM using both camera and laser range finder. The feature extracted should be stable and repeatable throughout the course of a navigating robot in order for it to be useful for SLAM.

This report is organized as follows. After related work in section II, We will discuss our approaches to the feature extraction with laser range finder as well as feature extraction with camera images in section III. Then we will evaluate the performance of our feature extraction results for both laser data and image data in section IV. Finally, we will conclude with section V and give directions to future improvements to our work.

II. Related work

Laser range scans provides accurate depth information about the environment. As described in Borges [1], laser range data can be processed and generate a line representation of the laser range data.

Vision has becoming an increasing popular topic in robotics. Cheaper camera cost, and faster CPU speed makes it possible to perform complex tasks in acceptable speeds.
The Vision Feature Extractor is a stepping stone to do Visual SLAM. The main motive of the Vision Feature Extractor is to identify a set of stable feature points, tag them with an index, and tracks it as the camera is moving. Plane homographies can be derived from stereo images, and can be used to calculate camera pose. The best works so far in Visual SLAM is no doubt Georg Klein [6] from Oxford University. His team has developed the MonoSLAM and PTAM, which are open-source. A brief overview on how the PTAM works is given below.

The PTAM has two main threads – one thread for grabbing image and do FAST feature extraction [4], and another thread for doing mapping and data association, as well as compute image homography planes. An image is grabbed from the camera, and a 5 level pyramid is constructed from the image. FAST feature extractor is performed on all levels of the pyramid and as many as 1000 points are obtained and stored in a keyframe. These points are tested for good edges using Shi-Tomashi algorithm, and they are ranked in order of non-decreasing scores. Those edges with good scores are randomized and K-Mean clustering was performed, which selects the patch with the best criteria. This patch is tracked in successive frames. The mapping thread is more computational intensive. It does homographic plane construction using stereo images from two consecutive frames. It also help to derive camera pose from the map and homographic plane.

III. Approach

A. Robot platform

The robot platform, Segway RMP 50 is being used for the MAGIC competition. There are five sensors on the robot: laser range finder, wheel encorder, IMU, GPS, and camera. These sensors communicate through sensor network to the on-robot computer. The computer, equipped with Core 2 Duo CPU, is running with Windows 7 operating system.

B. Laser range data

1) Breakpoint detector

We start off by using an adaptive laser range scan breakpoint detector to define disjoint line segments. This algorithm works by defining an imaginary circle for every laser points where its radius varies with range. Further range points would have a circle with a bigger radius, and shorter range points would have a smaller radius. If the neighboring point does not fall into the region of the circle, it is defined as a breakpoint as shown in the figure below.

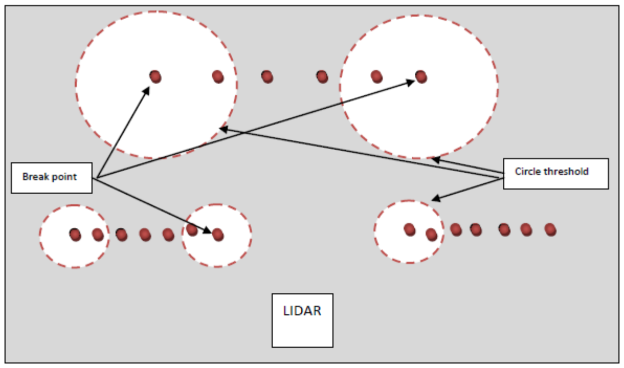

Fig.1 Diagram showing the idea of the breakpoint detector for laser range scans

2) *Line extractor*

A segment is defined as having two breakpoints at the end. However a segment may have more than one line, as the breakpoint detector only segments if it detects a rapid range change in the scan. An example would be having an "L shape" returned from a corner of two walls. The line segment algorithm takes the segments returned from the breakpoint detector and runs a divide and conquer algorithm.

A linear regression is used to compute the regression coefficient of the points in the line segment. All points with regression coefficient below a threshold are scanned in the segment and the point that gives the largest error will generate a new breakpoint. Then another run of linear regression for these breakpoints will create two line segments representing a corner. The regression is then repeated recursively and it outputs the equations of the detected lines of a single laser range scan.

3) *Corner extraction*

Corner features are other candidates for features to be used in SLAM. If a laser range scan is considered as a corner feature, the distance vector extended to its two nearest range scans must construct an angle that is less than a certain threshold. To filter out possible noise in the laser range scans, the distance vector extended to its two second nearest range scans must also have an angle that is less than such threshold. To extract corner features, dot-product is used to detect any laser range scans that satisfy the criteria for corner features.

Dot product: $\theta = \arccos\left(\frac{a \cdot b}{|a||b|}\right)$

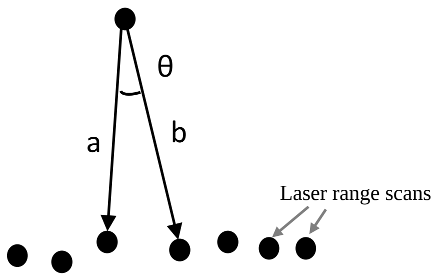

Fig. 2: Make use of dot product, a corner can be extracted as feature for laser range scans.

The output of the dot-product corner detector is not stable enough to be used on SLAM. There are detected corner features that can be difficult for the data association stage of SLAM. Therefore, we rejected detected corner features that are clustered within a certain threshold of range and bearing of the laser range scan.

C. *Camera images*

OpenCV 2.0 is used together with Visual C++ 2008 to implement the Vision Feature Extractor.

Initially the FAST feature extractor was implemented together with the image pyramids to extract raw feature points with the notion of a better optimized code. However there were difficulties to proceed with a robust feature extraction without proper software architecture. OpenCV offers some convenient functions to extract good feature points, and does a frame-to-frame feature points association using Optical Flow.

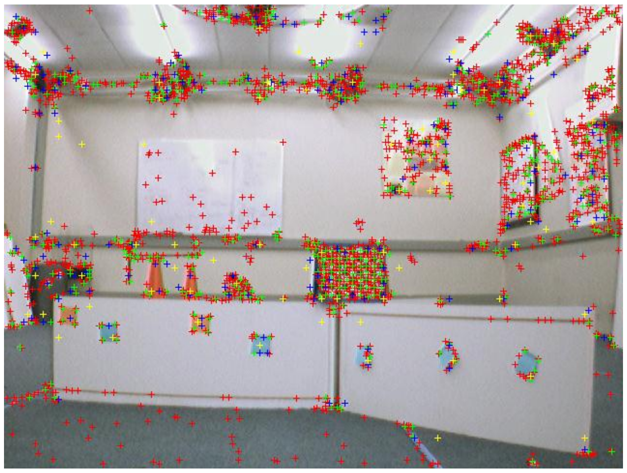
Fig 3: FAST feature extraction on a 5 level image pyramid. Different colours represent different pyramid levels. Red being the bottom level (highest resolution), followed by green, blue and yellow (lowest resolution).

1) *Algorithm*

1. An image is grabbed from the camera
2. If the map does not contain any feature points, 100 good features points are selected based on the Shi-Tomasi Corner detector.
3. K-Mean clustering is used to group the points into three clusters. The largest cluster of points is chosen and added to the map. Points are checked so that the same point will not get added twice.
4. On the next iteration, the map maker does an assessment of the points in the map. It determines the number of visible points in the map, and sends a flag to collect more feature points if the number of visible points falls below a threshold of 20 points. If there are enough visible points, the map maker consolidates the existing points from the map and passes them to the optical flow block to associate feature points of the previous and current images.
5. Feature points are stored in a data structure. Optical flow is used to find feature point correspondences of the previous and current images. Feature points which have correspondences are rewarded by the increment the in-liner count, while points which do not have any correspondences are blacklisted by increment the out-liner count.
6. The map maker controls the data association of the feature points. Points that have high out-liner count are

removed. A filter is used to detect and reject feature points that have noisy velocities. Good feature points will be stored into the map.

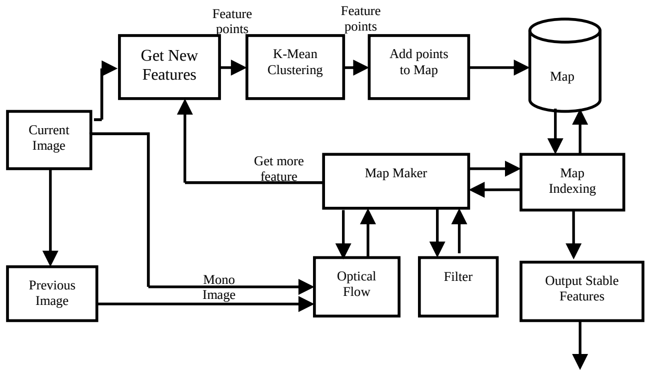

Fig 4: The work flow of the feature extractor using OpenCV library

2) *Filtering*

Wrong feature point association can happen if the robot is making a swift turn. We would like to detect and prevent wrong features from being updated in the map. A vertical displacement filter is implemented to remove wrongly associated features.

When the robot moves or makes a turn, the horizontal displacement of the feature points is at a higher magnitude than the vertical displacement, unless the robot is moving up a bump. At times, poor feature points association can cause the vertical displacement to jump. Thus we can make use of the vertical displacement to filter out bad feature association points.

The robot is driven around the room at 1m/s, making turns as it moves. A 5 minute video was recorded and feature points were extracted, in pixel coordinates, (u,v). We train the data set of the vertical displacement error pixel values using linear regression, and the vertical variance, δ_y was extracted. To separate good points from the bad feature points for the training data set, so we use a median filter of mask size 7 to filter out spikes caused by bad feature points.
After the variance of the vertical displacement was computed, we reject feature points by using a threshold on the vertical displacement error set to $3 \times \delta_y$.

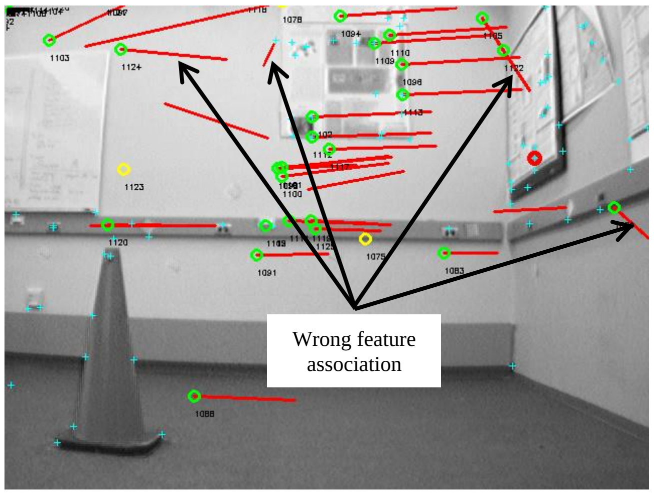

Fig. 6: Wrong feature associations that are pointed out.

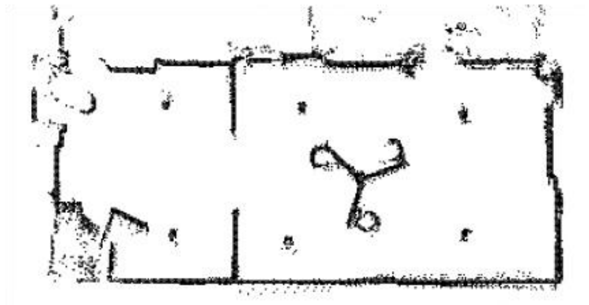

Fig. 5: Bird eye view of robot's environment.

IV. Experiment Result

A monocular color camera and a SICK laser range finder is mounted on the Segway robot, and the robot is exploring around a room of size roughly 5 by 12 meters, recording the images at 15 Hz. Posters, cones and sticky notes are placed around the room to more distinct features in the room.

A. *Laser range data*

1) *Breakpoint detector*

The detection rate of the breakpoint detector is fairly consistent. For a data sample of 200 scans, it has roughly 95% success rate of detection.

2) *Line detector*

The detector was able to detect lines. However, the lines are not stable enough to be landmarks. Therefore, a further improvement is necessary to the current detector.

3) *Corner detector*

The dot-product corner detector was implemented in MATLAB. In order to gauge the performance of the corner detector, we handpicked features that we think are the best stable features in the environment the robot is maneuver in. We compared the dot-product corner feature after the clustered features were denied with the handpicked features to show the performance of the dot-product corner detector. The mean of the ratio of detected corner with handpicked features was 0.4674 which shows that the dot-product feature extractor picks up unstable features about half of the time. The result shows the features extracted are not suitable for the SLAM.

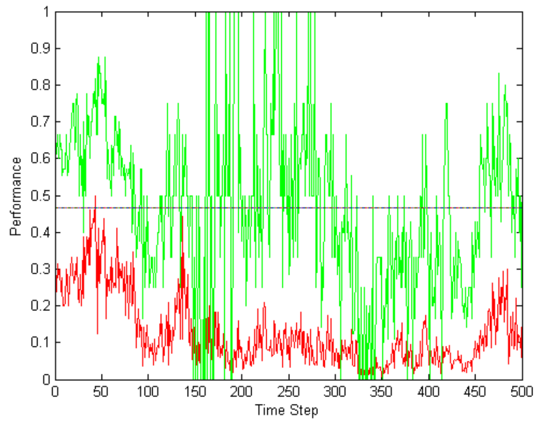

Fig. 7: Result of the dot-product feature extraction. Green line represents the result of clustered feature point denied. Red line represents the dot-product feature extraction before clustered feature denied.

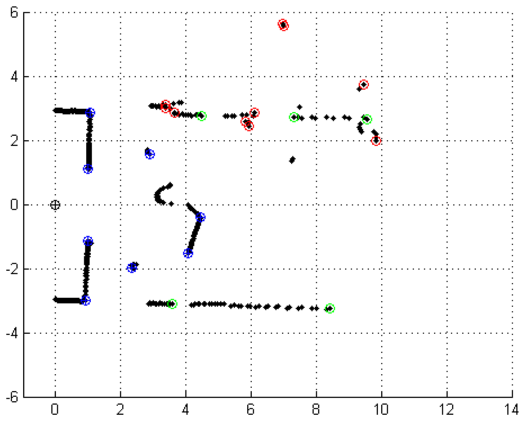

Fig. 8: Corner features detected from dot-product corner detection. Corner features that are clustered are shown in red. Corner features that are more stable features are shown in green. Corner features that are the best stable features (handpicked) are shown in blue.

B. Camera Images

The Vision Feature Extractor runs at 3 to 5 frames per second on an Intel Core i5 2.4 Ghz laptop computer. The performance could be further tweaked by careful planning of the software architecture utilizing multi-threading.

Objects which have high edge contrast are detected as features points. These feature points are tracked constantly in every frame. However there are some instant that objects with poor features, such as a white wall, were detected. These features usually appear and disappear between frames, and they do not get tracked constantly.

A video of the result of our implementation is submitted to the course email.

V. Conclusion

We have successfully extracted features from the camera images using OpenCV's function with corner extraction and optical flow. We also were able to extract features using dot-product corner extraction with the laser range finder. However, the features extracted were not stable and useful enough to be used on the SLAM. There are some possible future works to improve the result.

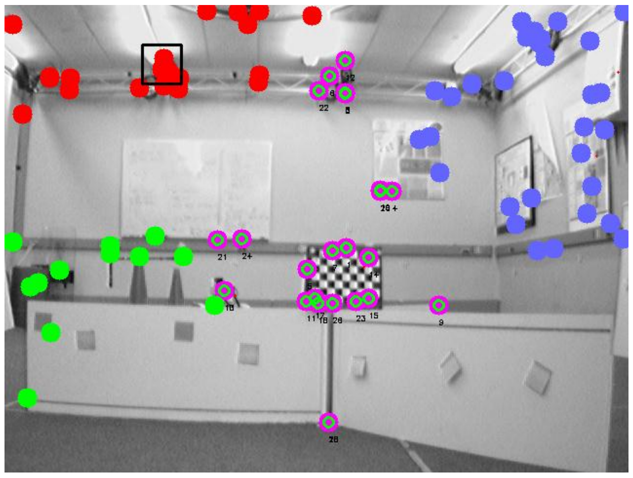

Fig. 10: K-Mean Clustering is performed on the image. Setting it to detect more clusters improves accuracy, but with higher computational cost. Currently, we set it to detect 4 clusters of points. The cluster with the most number of points is selected to be inserted into the map.

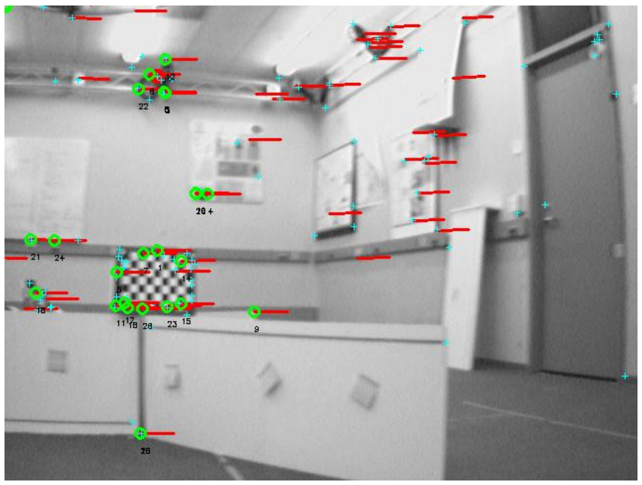

Fig. 9: Optical flow in action. The red line streaks represent the displacement of current and previous feature points using the Lucas-Kanade optical flow algorithm. The cyan crosses represent the feature points detected based on good edges using the Shi-Tomashi algorithm. Features that are deemed stable are represented by the green circles, which are indexed.

For vision,

- Make feature extraction and association more robust
- Make use of the output from the feature extractor
- Visual odometry

For laser range data,

- SVM approach
- Combining line and corner features to achieve more stable feature extraction

Acknowledgment

We would like to thank Professor Ashutosh to give us this opportunity to explore more about robotics and deepen our knowledge in the field of machine learning. We also like to thank Cornell University ASL to let us utilize the Segway robot for data collection and experimentation.

References

[1] G.A, Borges, and M.-J. Aldon. Line Extraction in 2D Range Images for Mobile Robotics. In: Journal of Intelligent & Robotic Systems, v. 40, n. 3, pp. 267-297, 2004.

[2] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics, MIT Press (2005).

[3] S. Riisgaard and M. Rufus Blas, "SLAM for Dummies", pp. 1-44, 2003.

[4] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in Computer Vision - ECCV 2006, (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951, pp. 430-443, of Lecture Notes in Computer Science, Springer, 2006.

[5] G. Klein and D. Murray: Parallel tracking and mapping for small AR workspaces. In: Proc Intl. Symposium on Mixed and Augmented Reality (ISMAR 2007), Nara (November 2007)

[6] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In ECCV, 2008.