Australian National University

LAB REPORT COVER SHEET

ANU College of Engineering and Computer Science
Australian National University
Canberra ACT 0200 Australia
www.anu.edu.au

+61 2 6125 5254

Submission and assessment is anonymous where appropriate and possible. Please do not write your name on this coversheet.

This coversheet must be attached to the front of your assessment when submitted in hard copy. If you have elected to submit in hard copy rather than Turnitin, you must provide copies of all references included in the assessment item.

All assessment items submitted in hard copy are due at 5pm unless otherwise specified in the course outline.

Student ID	U 6366102		
For group assignments, list each student's ID			
Course Code	ENGN 6423		
Course Name	Robotics		
Assignment number	Lab 4 – Part 2		
Assignment Topic			
Lecturer	Dr. Viorella Ila / Dr. Rob Mahony		
Tutor			
Tutorial (day and time)			
Word count		Due Date	05/11/2018
Date Submitted	04/11/18	Extension Granted	

I declare that this work:

- ✓ upholds the principles of academic integrity, as defined in the ANU Policy: Code of Practice for Student Academic Integrity;
- ✓ is original, except where collaboration (for example group work) has been authorised in writing by the course convener in the course outline and/or Wattle site;
- ✓ is produced for the purposes of this assessment task and has not been submitted for assessment in any other context, except where authorised in writing by the course convener;
- ✓ gives appropriate acknowledgement of the ideas, scholarship and intellectual property of others insofar as these have been used;
- ✓ in no part involves copying, cheating, collusion, fabrication, plagiarism or recycling

Table of Contents

Introduction

The objective of the paper is to build a global map in large scale through various sessions. The author uses graph-based approach that use nodes and links as poses and odometry respectively. The author points out there are solutions for single mapping sessions but when the mapping is held over multiple sessions in different locations (kidnapped robot problem) involves localization of the robot over the existing map (given that the robot starts off with a known location). The approach followed here is simple yet powerful, the graph links store transformation between two adjacent nodes with help of odometry information. Generating live maps demands high necessity of memory and constant communication with the hard drive/ database either on local / remote system.

Concepts proposed

Loop closure detection is the main concept proposed by the author which uses *Bayesian filter* which functions based on a threshold value above which, it is hypothesized that loop closure has been detected. To ensure the correctness of the loop closure, the algorithm also uses visual words (*SURF features*)[1] to compute the likelihood of the occurrence of the event. The graph optimization uses *Tree based network optimizer* where poses and transformations are passed in as constraints. Upon loop closure, the odometry errors are passed down to all links to make corrections to map. When the maps are created over various sessions, each optimized graph has their own roots with respective reference frames. *Memory management* involves sensory (SM), short term (STM), working (WM) and long-term memories (LTM)[2]. STM holds a specific number of nodes over time (say S), like queue. The oldest nodes are moved to WM where it will be considered for loop closure. WM operates based on time T which implies that data in WM gets transferred to LTM after time T. LTM is exclusive for storage of data alone. Any loop closure/ graph optimization occurs only in WM, i.e., when a loop closure is detected in WM some of the neighboring nodes from LTM are moved back to WM (*retrieval*)[2]. Nodes are held in WM based on *weight update* in STM which depends on time where the robot has spent the most time on. Basically, as long as a single node stays in WM, loop closure can be detected for data/ nodes in LTM.

The transfer of WM to LTM is done only to location with lowest weight. If multiple locations have same low weight, then oldest location is transferred to LTM. SM perceives the scene and sends it to STM. To improve the accuracy in loop closure detection, SURF features are extracted from a pretrained target environment (64-dimension vector), which are compared to the generated features and ratio is determined. If the ratio is low/ high, then the extracted feature is not considered/ considered respectively for the loop closure detection. Due to high dimensions of SURF features[1], a random four kd- trees is used to detect the new signatures, which is then used for location creation.

The *weight update* is held based on the similarity between different signatures of extracted features[3]. If the similarity is high between two locations (based on ratio of features), then the earlier features matrix is transferred to cleared new features matrix. The weights of both locations are increased, and the neighbors and loop closure links are redirected to the new location (new feature matrix) and the previous location is deleted from STM.

The role of *Bayesian filter* is to track the loop closure through probability that current location is one of the locations in WM[4]. For new locations (standard deviation is lower than mean), the likelihood is evaluated using mean over standard deviation ratio. If the likelihood is high then the robot is probably (0.9) in a new location at current iteration (t), given that there was no loop closure in previous iteration (t-1), which creates the belief for the next iteration (t+1). The remaining probability (0.1) lies to the event that loop closure occurs at the current iteration (t), given that no loop closure was detected in the earlier iteration (t-1). When loop closure occurred on a neighborhood location at t-1, the probability of loop closure at t is defined as a gaussian curve for a range of sixteen neighbors. The gaussian curve is followed to safely assume that some of the adjacent neighbors are not in WM, gaussian curve fits the probability sum as 0.9.

Discussions

The paper covered a lot of technical aspects of robotics, computer vision and computer science. The way of presentation of the work is appreciable and the references used are reliable and helpful in understanding the in-depth concepts proposed. Though author has provided his papers in his website with the right order to tread it the connection between new and earlier works could have been linked even more effectively. In the technical aspect, the author provides ample support to the researchers in implementing and improving his work. The memory management concepts discussed in the literature were difficult to catch in the first read due to lots of technical jargon exclusive to the computing field. Due to ample references, we were able to grasp the outline of the paper and understand the proposed concepts.

Goal of the project

The goals of the project include efficient mapping of the environment from RGB-D camera, planning and navigation in the generated map. The author aims to reduce the size of the generated map that is generated by moving the generated map to long term memory i.e., hard drive of the turtlebot computer.

The code given tries to accomplish above goal through

- ❖ Continuous collection of the bag of words (SURF Descriptors).
- ❖ Links from one image to another through bag of words and creates nodes for the graphs.
- ❖ After a threshold count of images, a new node is created, and the process is continued.
- ❖ These nodes are linked together, and the graph is optimized using the kd-trees approach.
- ➢ The optimized graphs are sent from WM to LTM for storage
- ➢ The stored graphs are retrieved when the loop closure is detected. (applicable to multi session mapping).
- Uses Bayesian filter for localization within the generated map
- Targets/ Navigation goals can be passed into maps through RVIZ and the path is planned in the existing map and the path is planned using Topological Path Planner (Dijkstra algorithm) and Metrical Path Planner.
- ❖ Map generation steps.
- ➢ Graph optimization steps.
- Navigation/ Localization steps

Implementing RTAB-Map and its results

While reading through the paper, the author encouraged to carry out the handheld mapping of the environment. We started out with it and mapped the environment which went smoothly. Later when we tried the same in turtlebot we faced issues of improper connectivity, odometer calibration errors, improper stitching of images to generate the map.

We calibrated the odometer again as taught in earlier lab sessions, calibrated the Kinect camera to ensure the proper generation of map. Next, after proper calibration when we carried out the same procedure, we learnt to map the environment smoothly rather than one full sweep 360-degree rotation to generate the map. Additionally, we also learnt that teleoperated robot generated provided more control over the situation rather than autonomous robot.

Initially, we generated maps only for single sessions for a small area (Fig 2.), to ensure proper map generation. Later, we conducted multi session mapping to generate Fig 3., which clearly shows the mapping of the apartment and the location of robot in the map (Cyan box).

We decided to implement the extension of RTB-Map which includes planned[3] navigation in generated map. We struggled in passing navigation goals to the robot. We learnt that the navigation goals need to be passed only when the turtlebot robot is out of network and remains standalone, which clearly solves the planned navigation. We tried to pass navigation goal through remote computer which apparently required map data in the remote computer. Due to the time constraints, we were not able to implement it, as it required conversion of map data in .db (database) format to .yaml format. It requires more time to understand on what parameters to passed into yaml file from database. The database included 8 tables of data of maps which can't be randomly assigned to the yaml file. It would have been better if author had explained how the maps are converted into the tables in database where simply assuming things won't help in understanding.

The attached video (Demo video.mp4) shows a short demo of planning and navigation of the robot in the generated map where in the first 5 seconds the navigation goal is passed, and the pose of the robot is passed as green arrow. The map in the video is different from Fig 2 and 3., which contains the obstacles/ objects in the environment as color patches.

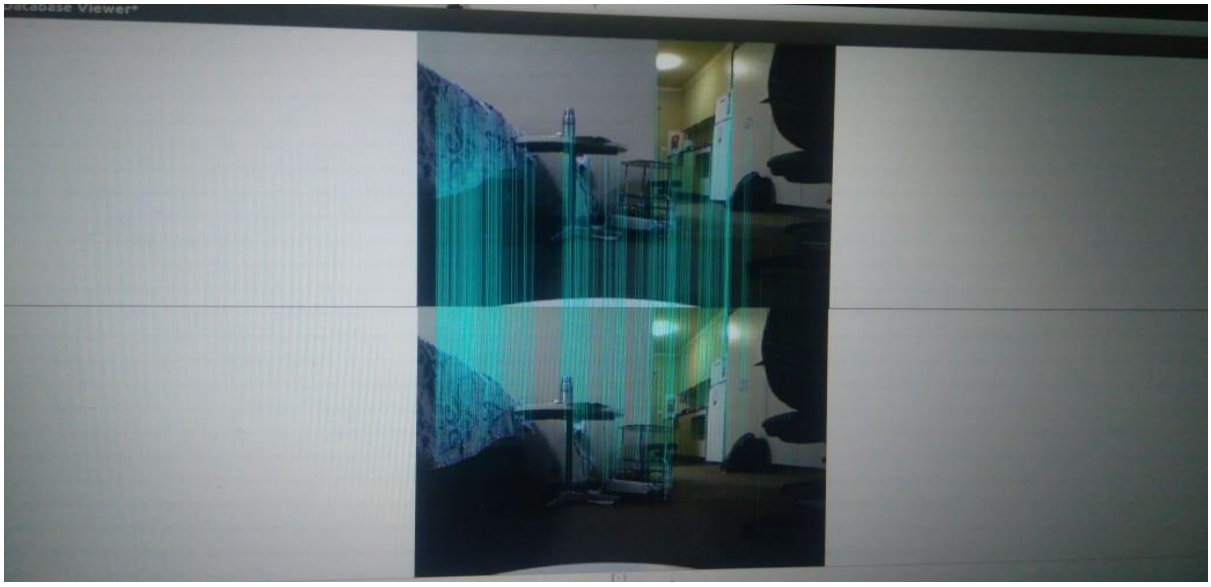

Fig 1. SURF Descriptors (Bag of visual words)

Fig 2. Session 1 results.

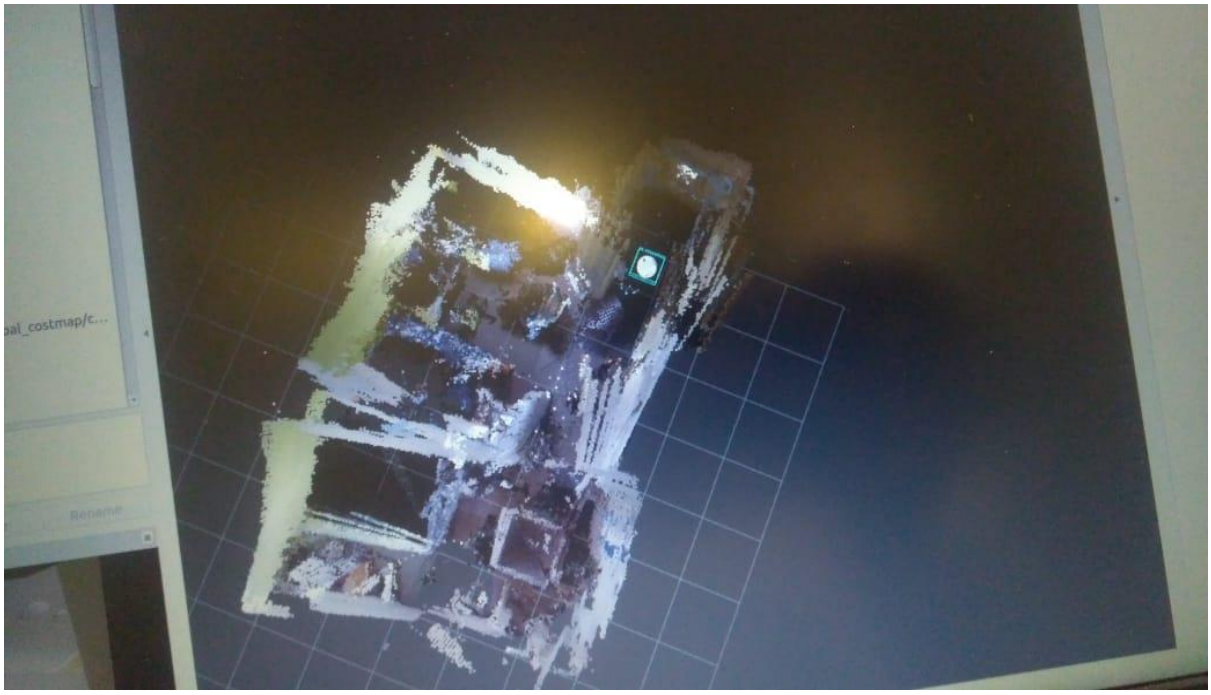

Fig 3. Integration after 3 sessions

Questions

Question 3:

Given:

Let x be the actual color of the car and y be the color observed by the witness

9 out of 10 taxis are green.

P(x=g) = 0.9

P(x=b) = 0.1

From statement, "Under dim lighting conditions, discrimination between blue and green is 75% reliable"

P (y=g | x = g) = 0.75

P (y=g | x = b) = 0.25

P (y=b | x = b) = 0.75

P (y=b | x = g) = 0.25

$$P\,(x = b \mid y = b) = \frac{P(x = b, y = b)}{P\,(y = b)}$$

$$P\,(x = b \mid y = b) = \frac{P(y = b \mid x = b)\,P\,(x = b)}{P(y = b, x = g) + P(y = b, x = b)}$$

$$P\,(x = b \mid y = b) = \frac{P(y = b \mid x = b)\,P\,(x = b)}{P(y = b \mid x = g)\,P\,(x = g) + P(y = b \mid x = b)\,P\,(x = b)}$$

$$P\,(x = b \mid y = b) = \frac{0.75 * 0.1}{0.25 * 0.9 + 0.75 * 0.1} = 0.25$$

Question 4:

```
p = [1/9 1/3 1/3 1/9 1/9];
for i =1:5
    f(i,:) = [p(i) * 0.1  p(i) * 0.8  p(i) * 0.1];
end
pfinal     =     [f(1,1)+f(4,3)+f(5,2)     f(1,2)+f(2,1)+f(5,3)
f(1,3)+f(2,2)+f(3,1) f(2,3)+f(3,2)+f(4,1) f(3,3)+f(4,2)+f(5,1)]
sum(pfinal)
```

The variable *p* is assigned with the current belief of the robot and probability of motion for each cell is being calculated in the *for* loop.[5]

The final probability matrix of the after the noisy action is computed in *pfinal*.

The final probability of the robot's motion is

0.1111	0.1333	0.3111	0.3111	0.1333

Question 5:

Steps of a Kalman Filter:

1. Calculation of predicted belief, mean and covariance are calculated from u_t, μ_{t-1}, Σ_{t-1}. R_t is the covariance of uncertainty due to state transition.

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

2. Incorporating the measurements from the sensors into the above equations through which they get transformed into the desired belief. K_t is the Kalman gain which specifies to what degree the measurement is incorporated into the new state estimate. The difference $z_t - C_t \bar{\mu}_t$ is known as the innovation term, where z_t is the measurement vector. Finally, new covariance term is calculated. Q_t is the measurement noise covariance.

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

The names of the belief after each step of the Kalman filter is represented by mean and covariance at any given time t (moments parameterization).

Steps to derive prior:[6]

$$\overline{bel}(x_t) = \eta \int \exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t)\right\} * \exp\left\{-\frac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1})\right\} dx_{t-1}$$

$$\overline{bel}(x_t) = \eta \int \exp\{-L_t\}\, dx_{t-1}$$

$$L_t = \frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t) + \frac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1})$$

The above equation is decomposed into two components $L_t = L_t(x_t) + L_t(x_{t-1}, x_t)$

$$L_t(x_t) = \frac{1}{2}(x_t - B_t u_t)^T R_t^{-1} (x_t - B_t u_t) + \frac{1}{2}(\mu_{t-1})^T \Sigma_{t-1}^{-1} (\mu_{t-1}) - \frac{1}{2}[A_t^T R_t^{-1} (x_t - B_t u_t) \Sigma_{t-1}^{-1} (\mu_{t-1})]^T (A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1})^{-1} [A_t^T R_t^{-1} (x_t - B_t u_t) \Sigma_{t-1}^{-1} (\mu_{t-1})]$$

The minimum of $L_t(x_t)$ is computed equating the first derivative to zero.

$$R_t + A_t \Sigma_{t-1} A_t^T (x_t - B_t u_t) = R_t^{-1} A_t (A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1})^{-1} \Sigma_{t-1}^{-1} (\mu_{t-1})$$

Solving for x_t, gives the mean of the belief $\overline{bel}(x_t)$,

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

The second derivative of $L(x_t)$ gives the covariance of the belief $\overline{bel}(x_t)$,

$$\bar{\Sigma}_t^{-1} = A_t \Sigma_{t-1} A_t^T + R_t^{-1}$$

Advantages of Kalman filter:

- Relatively low computational complexity $O(k^{2.4})$, where k is the dimension of measurement vector z_t.[6]
- Estimates solutions perfectly for linear models

Disadvantages of Kalman filter:

- It assumes that the state is normally distributed.
- It assumes both state and observation model equations are linear which is not true in real life situation.
- It does not work very well when the measurements are perfectly accurate i.e., D = 0 in Y = Cx +Dw (Output equation).

Kalman filter solves the kidnapped robot problem if the robot is kidnapped with in the generated map and both Markov algorithm and Kalman filter performs similarly in case of new location.

References

[1] B. Herbert, Tinne, Tuytelaars, Luc, Van, Gool, "SURF: Speeded Up Robust Features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346-359, 2008.

[2] Mathieu, Labb´e, Franc¸ois, Michaud, "Memory Management for Real-Time Appearance-Based Loop Closure Detection," presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011.

[3] Mathieu, Labb´e, Fran¸cois, Michaud, "Long-Term Online Multi-Session Graph-Based SPLAM with Memory Management," *Autonomous Robots - Springer*, vol. 42, no. 6, pp. 1133-1150, 2018.

[4] Mathieu, Labb´e, Fran¸cois, Michaud, "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734-745, 2013.

[5] Rob, Mahony, Viorella, Ila, "Lecture Notes / slides," *ENGN 6627 Robotics*,

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.