Project 6 SLAM Proposal

Group: Sterling Peet, Robert Grosse, Jack Morgan

4/19/2013

1 Project Description

We are comparing two distinct SLAM algorithms on a single dataset, looking for performance and quality differences between the two different approaches to this mapping algorithm, the second option for the assignment. We did this by taking two SLAM algorthms, TinySLAM and DP-SLAM, and performing the nessesary modifications to them to be able to run them on an identical dataset that was found on a public repository from MIT.

2 Description of SLAM Algorithms

- TinySLAM[1]:A mimimalist implementation of SLAM, it provides a localization library based around a particle filter. It's claim to fame is its size: under 200 lines of C-language code. Comes with no support or true documentation.
- DP-SLAM[2]: DP-Slam attempts to have simultanious localization and mapping without the use of landmarking. It claims to be accurate enough that no special loop closing algorthms are needed, thanks to the use of a particle filter over joint robot and map state. It is implemented in C++.

3 System Changes

All our code was written in C, using the GNU toolchain. It required no modification to compile correctly on our machine using the cygwin gcc toolchain (Windows 7, SP1, 64bit).

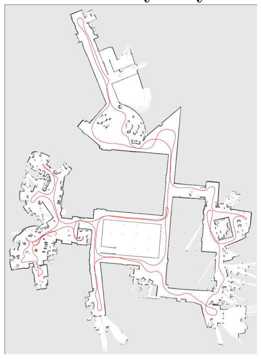
Our first major issue was converting our dataset from MIT into the format required by each algorthm. For TinySLAM, this meant modifying the file parser itself, as our dataset used floating point values, while TinySLAM works with integers. We converted the data by taking the float values, multiplying them by 1000, and truncating to integers.

[1]http://openslam.org/tinyslam.html
[2]http://openslam.org/dpslam.html

In the case of DP-SLAM, the dataset was converted into a format that matched what was listed in the corresponding DP-SLAM README file, and the file was edited to allow for a 360 degree sensor. Aditionally, the internal robot model was changed to fit our robot.

4 Dataset

MIT CSAIL 3rd Floor, Indoors[3]Date: 2005-12-17 robot: b21r laser: sick lms, 80m recorded by: Cyrill Stachniss



5 Evaluation

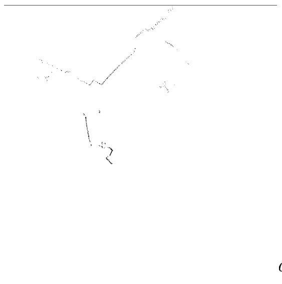
Based on the results, we saw definite differences between the two algorithms

TinySLAM seems to only render certain worldviews, while the data from DP-SLAM is more in the form of a blob, with some estimated walls. However, TinySLAM also seems to not render points that it has uncertanty about, and demonstates a degree of certanty with its mapping, as shown in the renderings.

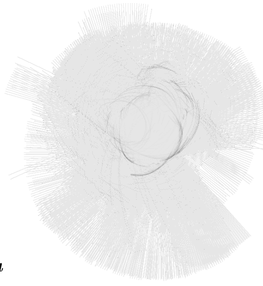
These two algorthms were run on a Windows 7 64bit computer, with 16GB of ram and a Intel Xeon CPU (X3430), clocked at 2.40GHz.

- DP-SLAM Memory Footprint : 766,108k (From Windows Task Manager) Runtime 89.7 Minutes. (Using Measure-Command in Windows Power Shell)
- tinySLAM Memory Footprint : 787,00k (From Windows Task Manager) Runtime 31.2 Minutes. (Using Measure-Command in Windows Power Shell)

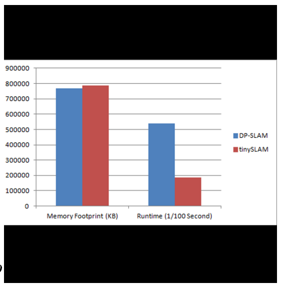
[3]http://cres.usc.edu/radishrepository/view-one.php?name=mit-csail-3rd-floor



[a]



[b]



[a] TinySLAM World Rendering
[b] DP-SLAM World Rendering

6 Discussion

We originally planned to compare the algorithm results by examining the number and angles of the inferred walls. If they were accurate enough, we could even do a per pixel comparison with each other and the ground truth image. Unfortunately, neither algorithm managed to reconstruct anything remtoely resembling the actual hall layout, as seen in the image that came with the dataset. This severly limits the amount of meaningful analysis we can do.

Most likely, the poor results were the result of an incompatibility in input format or incorrect parameters. The alternative, that the programs themselves were buggy, seems unlikely as another team reported being able to get actual results from DP-SLAM. For us, it was a miracle that we were able to get them to run at all. In fact, we had to switch algorithms halfway through, discarding TJFT-SLAM for exactly that reason.

We tried several variations, fiddling with the input files, and running it on different computers, but nothing seemed to help much. The amount of trial and error debugging we could do was severly limited by the fact that each run took over an hour. Furthermore, neither program had any real documentation.

Without the ability to get any quantitative data about the results to do a meaningful comparison of the algorithms, we are left comparing the performance and memory usage. While it is possible that a misunderstanding of the input format might lead to the algorithm operating on garbage data and encountering behavior and hence performance not typical of regular usage of the algorithm, this is still the best we have to go on. The memory usage of both algorithms was nearly identical surprisingly enough. This could be related to the fact that both algorithms use a particle filter based approach, but the number of particles and level of optimization would still play a large role in determining memory usage.

In the case of speed on the other hand, TinySLAM was the clear winner, almost three times as fast as DP-SLAM. A speed difference like this can be significant in real-time applications, expecially if the SLAM code is to be embedded in a robot as a part of a control system. As far as our understanding goes, the robot that was used to collect this data ran for 80 minutes, which puts into perspective the computation time required. Assuming a robot with comparable processing power dedicated to SLAM calculations, DP-SLAM still could not be used in real time, due to its 89.7 minute runtime, while tinySLAM, with its 31 minute runtime, is well within the computational limits.

7 References

1. Austin Eliazar, Ronald Parr: DP-SLAM 2.0: <http://www.cs.duke.edu/%7Eparr/dpslam2.pdf>

2. Bruno Steux and Oussama El Hamzaoui: tinySLAM : a SLAM Algorithm in less than 200 lines of C code, Accepted for the International Conference on Control, Automation, Robotics and Vision (ICARCV), 2010

3. Mark A. Paskin: Thin Junction Tree Filters for Simultaneous Localization and Mapping, In the Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2003 <http://paskin.org/pubs/Paskin2003a.pdf>