SLAM Performance on Embedded Robots

Undergraduate Student Research: Individual Project

Nima Shoghi Ghalehshahi
Georgia Tech
nimash@gatech.edu

Ramyad Hadidi
Georgia Tech
rhadidi@gatech.edu

Hyesoon Kim
Georgia Tech
hyesoon@gatech.edu

ABSTRACT

We explore whether it is possible to run the popular ORB-SLAM2 algorithm (simultaneous localization and mapping) in real-time on the Raspberry Pi 3B+ for use in embedded robots. We use a modified version of ORB-SLAM2 on the Pi and a laptop to measure the performance and accuracy of the algorithm on the EuRoC MAV dataset. We see similar accuracy between the two machines, but the Pi is about 10 times slower. Finally, we explore optimizations that can be applied to speed up execution on the Pi. We conclude that with our optimizations, we can speed up ORB-SLAM2 by about 5 times with minor impact on accuracy, allowing us to run ORB-SLAM2 in real-time.

INTRODUCTION

Simultaneous localization and mapping (SLAM) is the problem of, given an unknown world, mapping the world and localizing within that map. It is used by self-driving cars, uncrewed aerial vehicles (UAVs), autonomous underwater vehicles (AUVs), and vacuum cleaning robots, to name a few. In most of these use cases, the SLAM algorithm needs to run on embedded devices, such as the Pi. Our goal is to find the optimal conditions for running ORB-SLAM2 [5], a popular algorithm used for real-time SLAM with mono, stereo, and RGB-D camera inputs, on the Pi.

MEASUREMENT

In this paper, we use "performance" to refer to the time it takes to run the algorithm and "accuracy" to refer to the correctness of the output.

We use the following accuracy metrics:

1. Absolute Trajectory Error (ATE): ATE measures the holistic accuracy of the algorithm by comparing ground truth trajectories[1] to the predicted trajectories [6].

[1]The EuRoC MAV datasets provide ground truth pose captured using the Vicon motion capture system and the Leica MS50 laser tracker and scanner

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Table 1: Raspberry Pi Data.

	Stereo Camera			Mono Camera		
Dataset	ATE	RPE	$\sum t$ (s)	ATE	RPE	$\sum t$ (s)
MH01	0.038	4.76	1581.8	3.318	4.55	996.4
MH02	0.046	4.86	1300.6	0.365	5.07	777.0
MH03	0.044	4.09	1150.4	2.951	4.39	636.2
MH04	0.143	8.19	805.3	5.427	7.99	442.1
V1 01	0.087	2.34	1075.5	1.179	2.20	732.4
V1 02	0.064	2.44	635.0	1.262	2.06	360.4
V1 03	0.135	2.12	818.1	1.017	1.85	471.8

Table 2: Laptop Data.

	Stereo Camera			Mono Camera		
Dataset	ATE	RPE	$\sum t$ (s)	ATE	RPE	$\sum t$ (s)
MH01	0.037	4.83	202.0	0.538	5.40	92.9
MH02	0.049	4.86	145.4	2.822	4.44	76.5
MH03	0.042	4.08	138.2	3.445	4.68	65.4
MH04	0.064	8.20	85.6	6.185	8.26	40.7
V1 01	0.088	2.34	115.8	1.211	2.23	75.4
V1 02	0.066	2.44	68.8	0.905	2.06	35.8
V1 03	0.071	2.12	85.1	1.195	1.74	43.2

2. Relative Pose Error (RPE): RPE measures the amount of drift in the system by comparing relative transformations between successive poses for the actual and ground truth datasets[1] [6].

And the following performance metric, the total tracking time – $\sum t$ (seconds): $\sum t$ is the total number of seconds it takes to process all images, excluding any file loading or parsing time.

TESTING ENVIRONMENTS

The software is built, packaged, distributed, and executed using Docker. We use the following two test machines:

1. Raspberry Pi 3 B+ (4x Cortex-A53; 1 GB RAM; Raspbian)
2. OVERPOWERED Laptop 15+ (Intel i7-8750H; 32 GB RAM; Windows 10 Education)[2]

DATA AND ANALYSIS

The data collected follows below trends:

- Stereo camera inputs provide much better accuracy while taking 2× longer to process (see figures 1 and 2).
- The laptop is 8.8× faster with stereo camera inputs and 10.3× faster with mono camera inputs.
- Both machines show similar RPEs, indicating that there's not much difference in the drift.
- The Pi had a higher ATE on average. However, this can be attributed to a few outliers.

[2]Docker Desktop for Windows uses the Hyper-V hypervisor to set up a Linux virtual machine to run Docker. This process incurs some performance overhead.

Figure 1: $\sum t$ difference between our machines (higher is better).

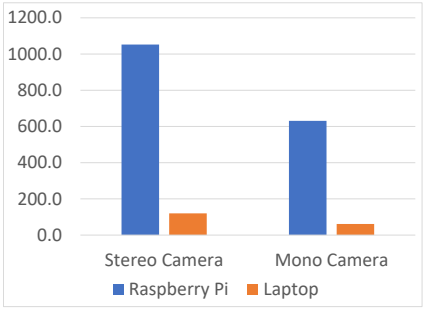

Figure 2: ATE and RPE difference between our machines (lower is better).

These trends indicate the Pi's hardware limitations degrade our performance but not our accuracy. The next section includes optimizations that increase our performance.

OPTIMIZATIONS

The following optimizations are applied to decrease the total tracking time $\sum t$:

- We tune ORB-SLAM2's ORB Extractor configuration:
  - Decreasing the number of features (*ORBextractor.nFeatures*) increases our performance with minor impacts on the accuracy. We witness a 1.75× increase in performance with negligable impact on accuracy.
  - Decreasing the depth of the orb extraction tree (*ORBextractor.nLevels*) has a 1.25× increase in performance. We increase the ORB scale factor (*ORBextractor.scaleFactor*) to make up for the lowered depth.
- If the input video contains high definition frames, we recommend downscaling to lower dimensions. Our research showed that resolutions around 640x480 provide a good balance of performance and accuracy.[3]
- Using lower FPS for the input image sequence is recommended for seamless live SLAM. We found 10 and 20 FPS to be sufficient for seamless live SLAM on stereo and mono camera inputs, respectively.

The following optimizations do not have an impact on $\sum t$, but have an impact on the total time it takes to run the algorithm:

- ORB_SLAM2 employs a bag of words place recognition system built on the DBoW2 library for loop detection and relocalization [4, 2]. The library's initial vocabulary loading takes a considerable amount of time on the Pi, as it parses the text file line-by-line to create the vocabulary tree. We solve this by creating the vocabulary tree once before any test runs and reusing the same object for all of our tests. Other possible solutions include storing the vocabulary tree's in-memory representation to the disk, removing the need for a decoding stage — the Cap'n Proto protocol uses this method [7].
- Streaming the dataset over a local area network (LAN) HTTP server, as opposed to loading the images from the SD card, drastically decreases the time it takes to read each frame on the Pi.[4] This optimization has no impact on the laptop's performance, which uses an NVMe SSD.

Our poster and presentation show the specific performance and accuracy information for each optimization.

CONCLUSION AND FUTURE RESEARCH

Our suggested optimizations increase the base performance of the ORB-SLAM2 algorithm by a factor of roughly 5×. With these optimizations, as well as proper calibration parameters [1], we can achieve real-time online localization and mapping. Our poster and presentation show our real-time online SLAM setup on the Pi running on live camera input, including details about camera calibration, optimizations and performance metrics, and accuracy. As potential further research, we will do the following:

- Prior research indicates that the performance overhead of running Docker containers on the Pi is negligible [3]. However, some Docker features, such as AUFS, incur additional overhead which will be measured.[5] We will also attempt running this algorithm without Docker.
- We will provide custom specialized DBoW2 vocabulary input [2] for specific environments that the robot operate in.
- We will use faster general purpose chips, such as the Pi 4.

REFERENCES

[1] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[2] Dorian Gálvez-López and J. D. Tardós. 2012. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics* 28, 5 (October 2012), 1188–1197. DOI: **http://dx.doi.org/10.1109/TRO.2012.2197158**

[3] Roberto Morabito. 2016. A performance evaluation of container technologies on Internet of Things devices. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 999–1000.

[4] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics* 31, 5 (2015), 1147–1163.

[5] Raul Mur-Artal and Juan D Tardós. 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics* 33, 5 (2017), 1255–1262.

[6] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *IROS'12s*. IEEE, 573–580.

[7] Kenton Varda. 2015. Cap'n Proto. (2015).

[3] The EuRoC dataset that was used in our research provides stereo input images of size 752x480.

[4] This optimization is not necessary when perfoming real-time online SLAM. In the case of real-time online SLAM, we can use OpenCV's *cv::VideoCapture* to read from the Linux camera device (e.g. */dev/video0*) directly.

[5] Our testing environment controlled these parameters to minimize the overhead incurred by Docker.