Monocular Camera Localization with Robo-Snail

Pujith Kachana
Graduate Mentor: James Choncholas
Faculty Advisor: Ada Gavrilovska
Georgia Institute of Technology
pkachana3@gatech.edu

Abstract

Localization is an increasingly important topic of discussion in robotics. Localization is the process of computing a robot's configuration in its environment relative to a predetermined map or landmark. While several empirically successful approaches to localization exist, the approaches are often complex and computationally expensive in real-world settings. This study therefore seeks to validate and benchmark the monocular Perspective-n-Point localization approach on a Raspberry Pi driven snail robot, first performing the computations locally and then expediting the process by securely off-loading the computations to a server.

1 Overview

Localization has widespread applications in modern robotics, supporting many high-level tasks such as motion-control and path-planning. It drives many robotic research areas including SLAM and navigation, paving the way for numerous recent innovations like autonomous driving and robotic movement (Huang and Dissanayake, 2016). However, despite the substantial body of literature and various proposed algorithms, localization still remains a complex task with no perfect solution. Depending on the nature of the task and the environment, localization may require complex sensor fusion, modeling uncertainty, and extensive calibration and fine-tuning. In the case of mobile robots, localization may even involve motion-modeling and constrained pose updates. Due to this complexity associated with localization computations, it is important to consider what algorithm best fits the robot's task and how the process can be optimized. One approach to optimize the performance of a localizing robot is to off-load the localization computations to a server, as proposed by James Choncholas. The focus of this study is to create a robot prototype that can locally compute its pose through PnP localization, and later off-load its computations to a server. This provides a benchmark for the speed-up provided by off-loaded localization, while creating an agent to perform the off-loading and the server networking.

1.1 Localization

Localization is fundamentally the estimation of an agent's pose with respect to its environment, and there are various pre-existing approaches that perform this task. Possible sensor measurements that can be used for localization include visual feature points, GPS measurements, inertial/encoder odometry, or any combination of these, just to name a few. Each combination of measurements is accompanied by its own set of algorithms and computations for pose estimation (Huang and Dissanayake, 2016). Among these numerous approaches, the localization method chosen for this project is Perspective-n-Point, as it is a well-established pose estimation algorithm and is supported by the secure server off-loading code base.

1.2 Perspective-n-Point

Perspective-n-Point (PnP) is a localization technique used to estimate a camera pose with respect to an image. The algorithm takes a set of correspondences (with the "n" denoting the number of correspondences) between 3D world points and their projection on the image plane and determines

the pose that would best fit these correspondences, assuming a pinhole camera model. The model can be expressed as $p = K[R|T]P$, where p is the homogeneous coordinates of a point in the image plane, K is the intrinsic parameters of the camera, R is the $3x3$ rotation matrix, T is the translation vector, and P is the homogeneous coordinates of the corresponding point in 3D world space (OpenCV, 2022b). Fully expanded, the pinhole camera model is as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where u and v are image coordinates, f_x, f_y, and γ are camera parameters of focal lengths and skew, u_0 and v_0 are are the coordinates of the principal point, $r_{x,y}$ corresponds to the rotation, t_x corresponds to the translation, and x, y, and z represent the point's corresponding world coordinate(OpenCV, 2022b). Assuming that the camera's intrinsic parameters are known, it can be seen from this equation that six degrees of freedom still remain, three for each rotation and translation. Therefore, PnP with at least three correspondences, or P3P, is the minimum viable form of PnP that can compute a pose estimate. However, to compute a pose with no ambiguity, at least six point correspondences are needed. Thus, we will attempt to maximize the number of corresponding points detected for PnP in this project.

2 Robo-Snail

The agent developed for this localization task is a snail shaped differential drive robot. The reasoning behind the snail design is that the speed of the Raspberry Pi's localization, and thus the speed of the robot's movements, is proportionate to that of a snail's. The shell also provides a natural enclosure for the electrical components, making for an elegant differential drive robot.

2.1 Structure

The 3D structure for the snail was designed and printed by James Choncholas. The structure is shown below. There is a large cavity in the back of the shell and the bottom of the snail to store the batteries, Raspberry Pi, and the motor driver (along with some googly eyes to complete the look).

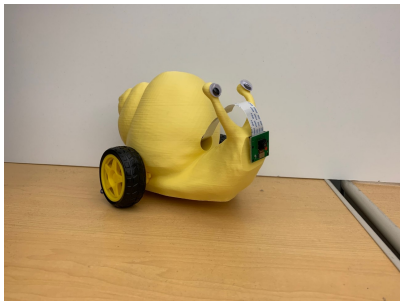

Figure 1: Snail Front View

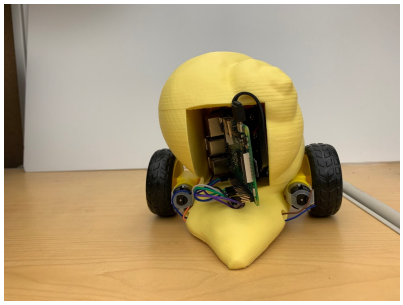

Figure 2: Snail Back View

The snail holds the Raspberry Pi and the various batteries in its shell and fits the motor driver in a small pocket on the underside. The parts are all held in place by mounting tape. The motors and wheels are screwed onto the sides.

2.2 Electronics

The snail employs a simple electronic system, using only a motor driver, two motors, a battery pack, and a Raspberry Pi 3B. The wiring is shown below.

The L298N motor driver draws a +12V input, provided by four AA batteries, and the Raspberry Pi draws a +5V, 2A input from the battery pack. The camera is inserted into the Pi's camera slot. Refer to the appendix for more details on the pinouts.

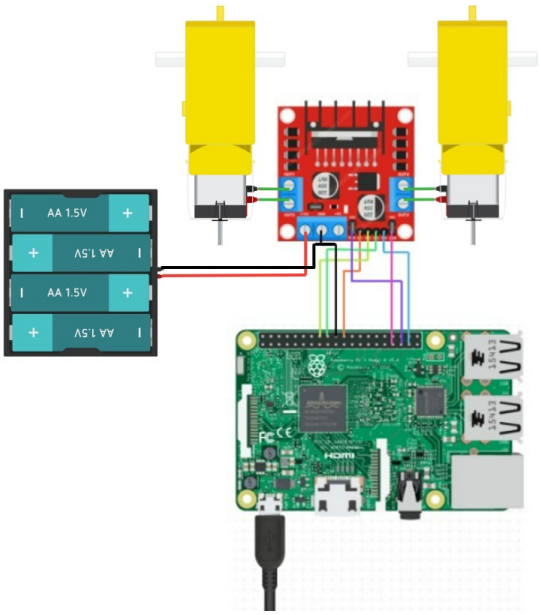

Figure 3: Snail circuitry

2.3 Software

The software architecture for the snail includes two separate modules to perform the localization and motor control, respectively. The localization module contains the Aruco board creation, camera calibration, and pose detection functionalities. It also includes the capability of off-loading the localization task to a server. The motor control module offers the fundamental differential drive functions, such as forward movement, backward movement, and rotation. The GitHub repository with the project code and resources for Aruco are included in the appendix.

3 Localization Procedure

3.1 Timeline

The Robo-snail is an ongoing project that started in January 2022. The first few weeks were spent defining the problem and detailing functionality requirements for the snail. It was decided that the snail should be able to support movement and PnP localization on the Raspberry Pi as a preliminary step, eventually allowing for secure server off-loading and more complex motion planning. February was dedicated to creating a prototype of the snail. Appropriate electrical components were deliberated and chosen, and a rough proof-of-concept differential drive model was created. The finer details of the project were decided around this time, including the use of fiducial markers and selection of a monocular camera as the only sensor. The final snail design was completed in March, concurrent with software development and literature review into localization methods for monocular camera robots. The localization repository was built in the first week of April, and the snail was fully assembled assembled shortly after. This marked the end of the Spring 2022 semester, and the project is planned to resume in Fall 2022.

3.2 Monocular Camera

The snail is equipped with a monocular camera which serves as its only sensor. Monocular cameras can be problematic when computing poses since they provide no notion of depth. In the pinhole camera model, every 2D point in the image maps to a line of candidate points in 3D world coordinates, making it impossible to accurately judge distance (MathWorks, 2022). A straightforward solution to this problem of depth would be to install a stereo camera or depth sensor. However, this approach is more expensive and increases hardware complexity. Monocular cameras are less expensive and more widely used, making a monocular camera solution more accessible and deployable. Although there is more initial processing overhead, it is possible to obtain distance approximations from a

monocular camera by calibrating on and obtaining poses from points of known distance in world coordinates. Using these points, a pose can be calculated for the snail by calculating the camera's intrinsic properties and performing PnP calculations. Fiducial markers from the Aruco library were used to facilitate this calibration and localization process for the snail camera.

3.3 Fiducial Markers

Fiducial markers are image markers used to serve as a reference point or unique id for visual systems. In the particular case of robot localication, fiducial markers can serve as easily and uniquely distinguishable feature points upon which a pose can be estimated. The Aruco library provides a simple interface with fiducial markers, generating squares embedded with a binary matrix (OpenCV, 2022a).

Figure 4: Aruco fiducial marker

The current implementation of the snail's localization uses a ChArUco board (a chess board composed of ArUco markers) to estimate a pose, as this would yield more feature points and correspondences for the PnP algorithm.

4 Results

The snail's monocular localization on a Raspeberry Pi with the ArUco marker implementation was tested and the performance was successful. The snail was able to identify the ArUco markers as feature points, extract the correspondences, and compute a pose relative to the markers using PnP. A snapshot of the visual output from the localization process is shown below.

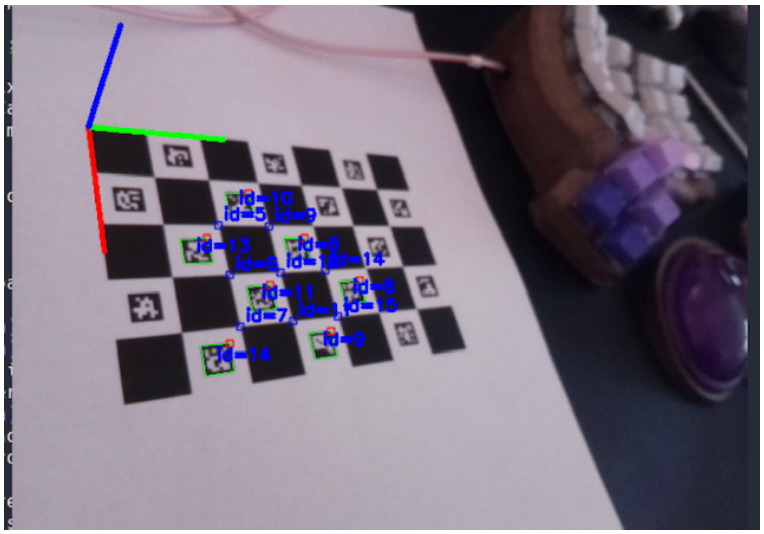

Figure 5: Fiducial detection and pose estimation

A shortcoming observed from the results is that the snail was only able to identify seven of the seventeen ArUco markers on the ChArUco board. While that is sufficient amount of correspondences to accurately compute a pose with PnP, it reveals a clear weakness in the snail's ability to detect feature points. Although the snail is capable of identifying the ArUco markers, the range is currently

limited by the camera quality. Plans to alleviate this issue in the future include using only two large Aruco markers to increase detection and range on camera.

5 Extensions

5.1 Secure Off-loaded Localization

The main purpose of this project is to develop an agent that can collect camera reading and transmit the feature points to a server to perform localization. Currently, the snail is capable of localizing on the Raspberry Pi using the ChAruco board, but this functionality has yet to be extended to a server. The next step for this project would be to securely network with a server to optimize the localization process, allowing the snail to move faster.

5.2 Position Based Visual Servoing Motion Control

Future plans for this project include embedding a motion control system onto the snail. The control system in consideration is position-based visual servoing (PBVS). Given that snail is a differential-drive robot (has two independent wheels on the same axis) and that we wish to use PnP localization for this project, PBVS is a suitable control system. PBVS uses the camera intrinsics and a target object of known 3D dimension (both of which can be calculated and controlled through the Aruco library) to transform the current reference frame of a differential-drive robot to a target reference frame with respect to the target object. This is done by calculating a rotation and translation matrix to drive the two wheels to trace a path to the target object (François Chaumette, 2006). Extending this motion control capability to the snail will make a complete and autonomous robot system, responding intelligently to its environment through its motion decision.

6 Conclusion

Localization is a slow and expensive process when performed on an embedded system, limiting its real-time capabilities. This projects aims to create a simple robot capable of visual localization with only a monocular camera, but, as shown by this study, there are various difficulties associated with performing this task locally. The camera needs to be calibrated and the scope of the feature points used to estimate a pose needs to limited to those of known distances. Even after enforcing these constraints, the result is generally a slow, snail-like robotic system. On such compact embedded robots, performance may degrade further if a motion control system is introduced. Increasing the computing power of the robot is a viable, albeit expensive and non-scalable solution to this issue that results in a larger robot, inducing mechanical and hardware complexity. A more elegant solution would be to off-load the localization task to a server, allowing the robot to simply act as an agent relaying sensory data and acting upon motion plans. This would in turn lead to a more efficient and ergonomic localizing robot design. This off-loading approach will be further studied and benchmarked in the future stages of this project.

References

François Chaumette, S. H. (2006). Visual servo control, part i: Basic approaches. *IEEE Robotics and Automation Magazine, Institute of Electrical and Electronics Engineers*.

Huang, S. and Dissanayake, G. (2016). Robot Localization: An Introduction. *Wiley Encyclopedia of Electrical and Electronics Engineering*.

MathWorks (2022). Calibrate a monocular camera.

OpenCV (2022a). Detection of aruco markers.

OpenCV (2022b). Perspective-n-point (pnp) pose computation.

A Appendix

A.1 Notes of Study

Notes:
https://docs.google.com/document/d/1EKmWVRrAaCyrqVSVwfYr__fraXVJz-KSU7UQabZMRq8/edit?usp=sharing

A.2 Source Code/Resources

Project Code: https://github.com/james-choncholas/snail
Aruco Library: https://docs.opencv.org/4.x/d9/d6a/group__aruco.html
Aruco Tutorials: https://docs.opencv.org/4.x/d9/d6d/tutorial_table_of_content_aruco.html

A.3 Project Supplies

- Raspberry Pi (3B used for this project)
- Pi Camera
- Raspberry-Pi Battery Pack
- L298N Motor Driver
- DC-Motors x2
- Wheels x2
- Jumper Wires
- 4 AA Battery Holder
- AA Batteries x4
- Access to 3D Printer