Particle Filter Localization Using Tactile Sensors For Robotic Applications In Challenging Environments

Hassan Jardali
Lebanese American University
hasssan.jardali@lau.edu

Noel Maalouf
Lebanese American University
noel.maalouf@lau.edu.lb

***Abstract*—Robots rely heavily on LiDar and vision sensors to achieve Simultaneous Localization and Mapping (SLAM). However, these robots can have missions in hazardous and challenging environments where perception sensors might malfunction. In this project, we aim to localize CHAMP, a quadrupedal robot, in a known map based on force/torque sensors attached to the robot's feet. The particle filter (Monte-Carlo Filter) is used as the localization algorithm. The proposed localization scheme is also simulated on TurtleBot, where eight bumpers are added to achieve tactile sensing.**

***Index Terms*—Localization, Particle Filter, Quadruped Robot, Tactile Sensors, Gazebo**

I. Introduction

Recent advancements in electronics and actuation technological tools have allowed legged robots to be used in various applications such as inspections or in emergency-aid missions. The working environments of these missions, such as mines or industrial plants, can have high density of dust and dirt that can affect the robot's perception sensors. These sensors, mainly Lidars and cameras, can easily be damaged by air-dust or water vapor, and their efficiency will drastically drop in dark areas. These robots need to have a high level of autonomous navigation capabilities to complete successfully the missions entitled to them. To achieve autonomous navigation, the robot needs to localize itself in a pre-defined or scanned map.

In this paper, we present a localization approach for quadruped robots, independent of perception sensors, and based solely on force contact sensors and odometry data.

II. Background

For us to understand more the problem statement, we need to have a background in various topics.

A. Quadruped Robots

Robots are usually classified into three categories: wheeled, legged and tracked robots. Legged robots have numerous advantages over other types, especially in manoeuvrability, transverse-ability, and navigation over obstacles [3]. Four-legged robots offer the best mobility and stability locomotion, and they are easily controlled and maintained as compared to other legged robots [3]. Quadruped robots are used in various fields such as military application, space exploration, industrial monitoring, in disasters' emergency areas and many more.

This paper focuses on robotic applications for quadruped robots in harsh environments.

B. Autonomous Navigation

To achieve autonomous navigation, the robot needs to first have its own locomotion and body control algorithms. Then, the robot will start to move. While moving, the robot will start to use perception sensors, mainly Lidar, to scan the environment. While scanning, the robot will start to build the map virtually. Simultaneously, the robot will localize itself within this map [6]. The latter algorithm is called SLAM: Simultaneous Localization and Mapping. When the robot is localized, the operator can command the robot to move to a specific location in the map. The robot will use motion and trajectory planning algorithms to complete the mission.

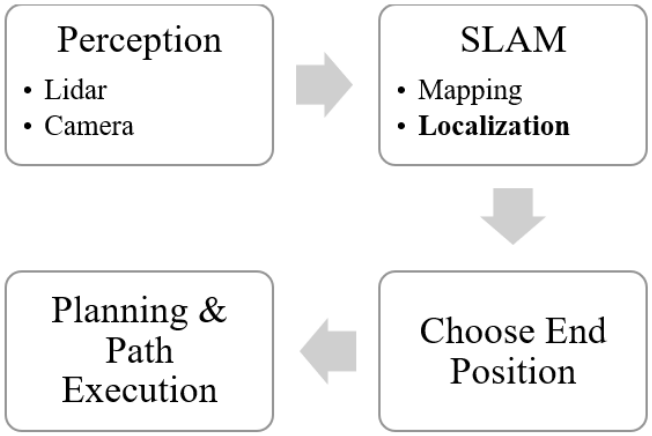


Fig. 1. Autonomous navigation steps

When the map is built, the robot needs only to detect features using its perception sensors, Lidar, or camera to localize itself.

The probabilistic map-based localization methods are mainly based on two different approaches: Particle Filter and Kalman Filter. Particle filter is explained in the next section.

C. The Localization Problem and Particle Filter

Particle Filter, or also known as Monte-Carlo Localization Filter, is an algorithm used to solve the localization problem.

Assuming the robot has already the map, defined as walls and obstacles with their respective locations, and assuming also the initial point is known, the robot can keep track of its location using odometry data. The odometry data, in 2D map, includes the X and Y positions and Yaw angle. However, realistically, the encoders are not reliable since minor inaccuracies in the odometry data can lead to a divergence in the tracking of the real position of the robot. Therefore, the robot needs to use data from exteroceptive sensors to keep the convergence towards the true position of the robot. These sensors usually include vision and LiDar sensors.

One probabilistic approach used to solve this problem is Markov localization. Markov localization tracks the robot's belief state using an arbitrary probability density function to represent the robot's position [2].

In order to reduce the computational complexity of Markov approach, a randomized sampling method was implemented to Markov filter. This filter is called Particle filter, or Monte Carlo localization filter [2].

In Markov, the filter will track every possible position in the map, however, in particle filter, random positions are considered, these positions (X, Y and O) are called particles. Every particle has a weight w which indicates the probability of a particle being at the true position of the robot.

III. Related Work

A. Tactile terrain classification localization

Buchanan et al. [1] used contact sensors to classify the terrain after analysing the data with a neural network, then this data is fed to a sequential Monte Carlo Localization Filter, a descent of the Particle Filter. The neural network structure consists of convolutional, recursive, and fully connected blocks. Their method proved an average position error of 20 cm over various terrain and geometries.

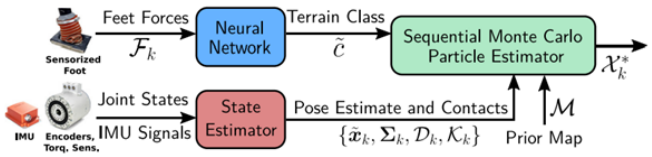


Fig. 2. Buchanan et al. approach

B. Haptic localization of legged robots

Chitta et al [4] proposed method relies only on joint encoders and IMU data and allows the robot to localize itself even though the initial position is uncertain. The method is the result of a comparison of a model-based calculations of the possible poses in a known map, with the current information as observed from joint angles and odometry data.

Schwendner et al. [5] localization approach achieved an average position error of 39 cm by using haptic localization on a wheeled robot. This method uses embodied data for localization, map building and SLAM. The wheels provide multiple contact points, that helps in building the pose estimate of the robot. However, perception sensors are also used in this method.

IV. Problem Statement and Proposed Method

As already stated, the localization problem can be solved by using LiDar sensors. In this research, we present an approach to solve the localization independently of LiDar or camera sensors since these sensors are not accurately functional in challenging environments.

Our proposed method is to attach force-contact sensors to the feet of the robot. These sensors will output the force and the torque exerted on the robot's foot.

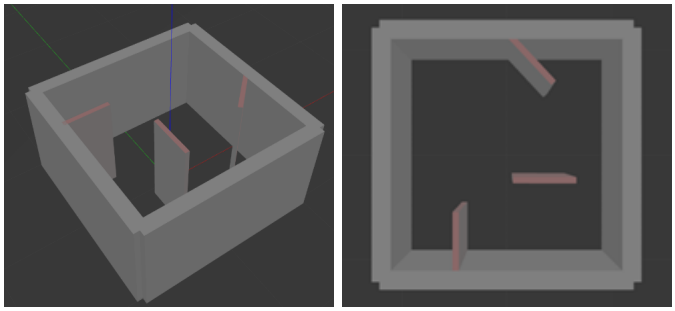
The robot will move its legs and touch the obstacles or walls in its reachable area, performing a probing action. Hence, the robot will use the particle filter along the data needed, odometry and probing data, to solve the localization problem.

V. Implementation

The approach suggested by this paper was tested on two simulated robots: TurteBot3 and CHAMP. Gazebo, a ROS-based simulator, was used in both experiments. At the end of this section, the results will be presented.

A. Map

To simplify the experiment, a simple map was built on Gazebo. The latter includes four walls forming a room, in addition to three walls serving as obstacles inside the room. The room is 2 by 2 meters. Figure 3 show the map in Gazebo.



(a) (b)

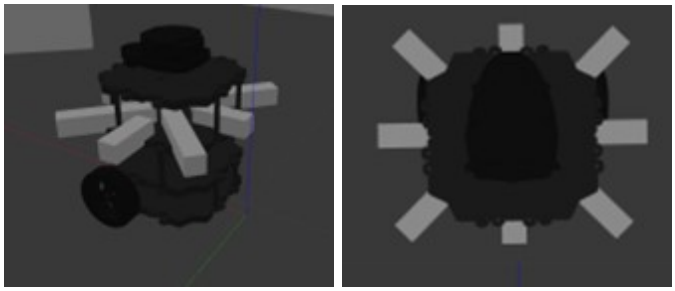
Fig. 3. Maps in Gazebo

B. TurtleBot

Despite not being a legged robot, we started our implementation on TurtleBot robot. Note: The filter can be used with more dimensions (in 6D), however, to simplify the analysis, we only used 3D (x, y and yaw).

1) Bumpers: Since the approach is based on force – contact sensors, bumpers, in cubic shape, were added to TurtleBot. A total of eight bumpers were mounted uniformly on the robot. Figure 4 shows TurtleBot with the bumpers.

The following configuration allows the robot to perceive the bumps from all the sides of the robot.



(a) (b)

Fig. 4. TurtleBot in Gazebo

2) Auto Navigation Feature: A simple algorithm was coded to assist the robot in navigation during the simulation. The code will check if the robot bumps into a wall, and if it does, the robot will rotate and move to the opposite direction. This code was based on the tele-operation feature of TurtleBot.

3) Extract Data Feature: A python code was also coded in order to automatically save the data extracted. The code is based on NumPy and panda libraries. Odometry data includes the x and y position and orientation of the robot, in addition to the bumpers data. The data is retrieved from ROS topics.

4) Simulation: The robot was first simulated in Gazebo for 220 seconds, with bumps on average every 20 seconds. The data was extracted and saved to be fed to the particle filter.

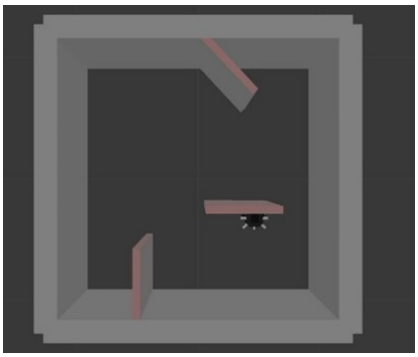


Fig. 5. TurtleBot during the simulation

5) Results: After simulating the robot, the data was fed to the particle filter. The particle filter used was based on Lukovic's implementation [7]. Figure 8 show the particle filter results.

MLE: Most likely estimate, the particle with the highest weight.

- The particles were first distributed over the complete map
- After the first bump, the particles concentrated in three different areas
- After the 4th bump, the particles converged towards the real robot location
- After the 6th bump, most of the particles converges towards the real robot location, however, the location of the MLE (red dot) differs around 20 cm of the real robot pose (green dot)

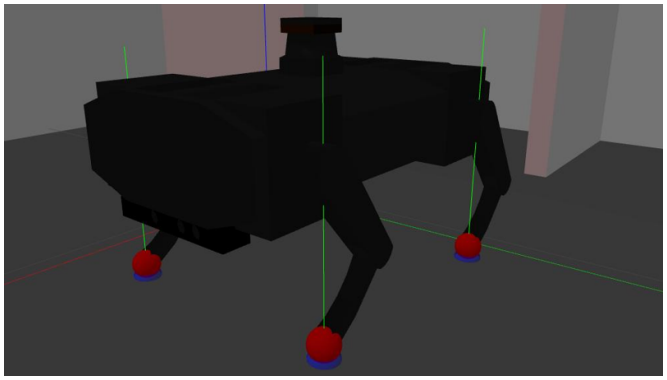


Fig. 6. CHAMP Robot

6) Notes:

- The particles converged successfully
- Note that only few simulations were performed with TurtleBot since the goal of this research is to implement the proposed localization approach on a legged robot.

C. CHAMP

CHAMP is an open-source development framework for building quadrupedal robots and developing control algorithms. The control framework is based on MIT cheetah robot control algorithms [8].

The previous map was used in this simulation. The extract-data feature from TurtleBot was also used with CHAMP.

1) Force-Contact Sensors: In Gazebo, bumpers' sensors, or force-torque sensors, were added to the feet links of the robot. The robot has a total of four bumpers' sensors.

These sensors are red coloured as shown in the figures 7 and 8.

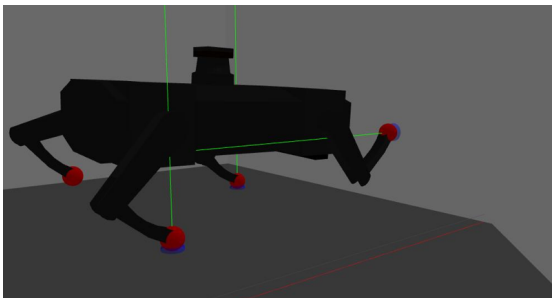


Fig. 7. CHAMP Robot while probing

2) Probing: The probing action was implemented by modifying the controller code in CHAMP repository. A subscriber (ROS) was added, then using the inverse kinematics function in the controller, the foot will move to the specified position. The four legs can probe (RF, RH, LF and LH).

3) Simulations: A simulation of 400 seconds was simulated. The navigation of the robot was achieved through the tele-operation feature of CHAMP. The probing action was manually commanded.

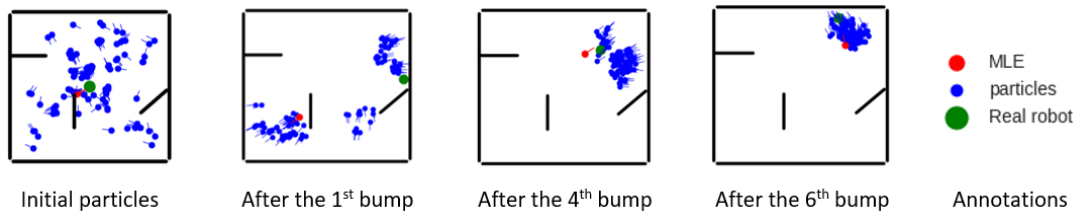


Fig. 8. Results of TurtleBot simulation

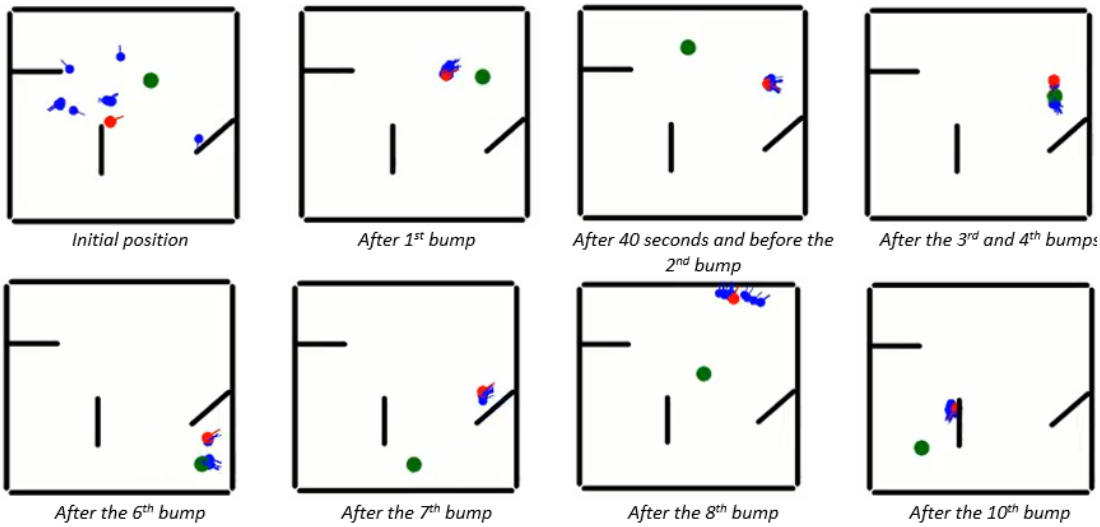


Fig. 9. Results of CHAMP simulation

Simulation	Nb of Particles	Nb of Steps (step time = 0.5s)	Std. Deviation (X and Y Pos)	Localised (MLE error <0.2m)
1	15	550	0.01	From step 142 (3rd bump) till step 290 (7th bump)
2	30	550	0.01	Steps 22 till 80 (after 1st bump) then from step 142 (3rd bump) till step 340 (8th bump)
3	10	180	0.03	Only 2nd bump
4	150	180	0.03	only 3rd and 4th bumps
5	150	250	0.025	only 3rd and 4th bumps

Fig. 10. CHAMP robot simulations summary

4) Results: The results shown in figure 9 are for a particle filter with the following parameters: number of particles: 15 and number of steps: 550 and variance for x and y positions: 0.01.

As shown in figure 9, the particle filter succeeded in localizing the robot till the 7th bump, however, after the 8th bump the error start to diverge and the particles became concentrated in a completely different area.

The table shown in figure 10 summarizes the results of the particle filter simulations.

The best results were retrieved from the 2nd simulation where the number of particles was increased, the number of steps is the highest and the standard deviation is the lowest.

CONCLUSION

D. Findings

After the implementation of our approach using ROS and Gazebo, we found that the proposed filter has succeeded in localizing the robot, however, after several bumps, the filter is losing track of the true pose.

E. Future work

In the experiment covered, few limitations can be tackled to have better results. These limitations include the map, which does not include many unique features. In addition, more simulations in more diverse environments can lead to better understanding of the problem of divergence.

Furthermore, the sensor used can be manipulated to have more features extracted, and more information about the obstacles sensed, and therefore the efficiency of the filter can be increased.

REFERENCES

[1] Buchanan, R., Bednarek, J., Camurri, M. et al. Navigating by touch: haptic Monte Carlo localization via geometric sensing and terrain classification. Auton Robot 45, 843–857 (2021). https://doi.org/10.1007/s10514-021-10013-w

[2] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. 2011. Introduction to Autonomous Mobile Robots (2nd. ed.). The MIT Press.

[3] Priyaranjan Biswal, Prases K. Mohanty, Development of quadruped walking robots: A review, Ain Shams Engineering Journal, Volume 12, Issue 2, 2021, Pages 2017-2031, ISSN 2090-4479, https://doi.org/10.1016/j.asej.2020.11.005

[4] Chitta et. Al work: Chitta, S., Vernaza, P., Geykhman, R., & Lee, D. (2007). Proprioceptive localization for a quadrupedal robot on known terrain. In IEEE international conference on robotics and automation (ICRA) (pp. 4582–4587). https://doi.org/10.1109/ROBOT.2007.364185
[5] Schwendner, J., Joyeux, S., & Kirchner, F. (2014). Using embodied data for localization and mapping. Journal of Field Robotics, 31(2), 263–295. https://doi.org/10.1002/rob.21489.y
[6] Douglas, B., Autonomous Navigation Video Series (2020), Mathwork.
[7] Lucovic, A., mobile robot localization using particle filter, github.com/lukovicaleksa/mobile-robot-localization-using-particle-filter
[8] CHAMP library, https://github.com/chvmp/champ