TECHNISCHE UNIVERSITÄT WIEN

Autonomous Racing Cars
191.119 (VU 4,0) Semester: 2022S

Lab 4: Mapping and Localization

2022-04-19

Preface

Read all the instructions below carefully before you start working on the assignment, and before you make a submission. All sources of material and ressources must be properly cited (this also includes datasheets).

- Completeness of solution: A complete solution of a task also includes knowledge about the theory behind.
- Exercises are to be solved in teams. All team members must be indicated on the submission protocol. However, every team member must be able to explain the handed in solution. Grading is on an individual basis. Upload your solution (one per team) in TUWEL until 2022-05-03 23:59.
- For this assignment there is no exercise interview. However if submissions are unclear, students might be invited for exercise interviews.
- One team will additionally present their work at the *Lab 4 presentation*.

Learning outcomes

The following fundamentals should be understood by the students upon completion of this lab:

- Create maps of unknown environments with SLAM or other methods.
- Use and tune localization algorithmns on a map and with different sensor inputs.
- Document your work clearly and in a reproducible way.

Deliverables and Submission

- Write a exercise report and submit it as PDF file in TUWEL until 2022-05-03 23:59. Use the provided latex template and do not forget to fill in the parts marked with "TODO". The anonymous version of the lab report (all pages except the first one with personal data) and the submitted source code archive will be shared with all the teams after the submission deadline. In any case the report must meet an adequate level for layout and readability that is appropriate for the academic context. It needs to be detailed enough, so that another team could reproduce your work without additional information.
- Submit a ZIP-Archive with all the relevant source code in TUWEL until the same deadline.

Transparency of Contribution

Describe in your submission protocol briefly how you worked together. How did you structure your work distribution and collaboration? Who contributed how much effort to which part of the work? (If one, or more team members, are not able to work on this assignment, you must also transparently state this here.)
(Please do not understand this preamble wrong to somehow exaggerate your contribution estimation: If you are working together well in your team, it should anyway be no problem to briefly describe how you worked together.)

1 Creating a Map

In the lectures we talked about different types of controllers for autonomous racing. Some (e.g. bang-bang, PID) work with rather simple inputs to create an immediate action. These algorithms do not need a map and can therefore not incorporate more information such as the direction of the track ahead. In contrast other algorithms need the (estimated) pose on a given map to do some planning and then derive the action based on this information together with the current sensor inputs.
To work with these algorithms in the upcoming labs we therefore need a map of the track (which might be an unknown environment) and thereafter some method to localize our car on this map. In the first step you will deal with the map generation.

1.1 SLAM

Wikipedia describes SLAM nicely in one sentence: "Simultaneous localization and mapping (SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it." [Wik22]. The lecture slides provide some information on the SLAM algorithm. There is also a lot of scientific literature about SLAM that you might read if you are interesed in more detail about the functionality and/or math behind it. For the context of the lab we are going to use a ROS-package that already implements the SLAM algorithm:

a.) Add a SLAM implementation to your ROS workspace. You might use *gmapping* [Gio22], [Sta22b] or *Google cartographer* [Aut22] or any other appropriate package. Drive your car with the PID controller node (you might decrease the speed) around the track and record the map. (Alternatively you can also drive manually.) Adjust the parameters of the SLAM-package to get the best results. Discuss the parameter modifications and their implications.

b.) Complete the task (1.1.a.) on at least three different tracks in the simulator. Show and discuss the results in your protocol. If your selected SLAM-package provides any internal information (e.g. transforms between submaps, etc.) visualize them as well and include it in the protocol. How does the resulting map change if you drive 1, 2 or 3 full laps on the track while mapping? Does the quality of the result depend on the starting position? What information (e.g. sensor data) is processed by your chosen SLAM-package (give arguments how you conclude this)?

c.) Use `map_saver` [Sta22c] to save the maps from task (1.1.b.). Include them in the protocol and compare them to the original map which was used by the simulator. Are there differences? Are there some specific sections that differ? Do you think these maps are good enough to use them for navigation algorithmns?

d.) Record a ROS bag [Sta22d] while driving in the simulator on one track for 3 full laps without the SLAM package running. Afterwards play the ROS bag and run the SLAM package to create the map offline. Use `map_saver` [Sta22c] to save the map. Describe the workflow and discuss if there are any points to take care of.

e.) Find a way how to automatically cut ROS bags in smaller sections. Do this for the data which was recorded at task (1.1.d.). Run the SLAM package on these smaller ROS bags and see if the map is different than the map based on the full ROS bag.

f.) Run the SLAM-package on the real hardware race car. Provide the answers to the questions as in task (1.1.a.). It might be useful to apply task (1.1.d.) accordingly to give you more flexibility without the need to do all your work on-site.

g.) Where there modifications (e.g. of parameters) needed to adjust your solution of task (1.1.a.) to the task (1.1.f.)? If yes, give details and possible reasons.

2 Localization

The task of localization is now to determine the pose of the mobile object (robot) on a given map.

2.1 Particle Filter

Particle filters are a well known method for localization in the field of robotics. The lecture slides provide some information on particle filtering and there is also a lot of literature on particle filters around. In this task you will use a ROS-package that implements a particle filter to localize your robot on the map:

a.) Add a particle filter implementation to your ROS workspace. You might use *AMCL* [Sta22a] or any other appropriate package. Adjust the parameters of the particle filter to get the best results. Discuss the parameter modifications and their implications.

b.) Do tests for task (2.1.a.) on at least two different tracks in the simulator. For both tracks first use the given ground-truth map as used in the simulator and then use the recorded map from task (1.1.c.). Show and discuss the results in your protocol. If your selected particle filter implementation provides any internal information (e.g. point-cloud, etc.) visualize them as well and include it in the protocol. What information (e.g. sensor data) is processed by your chosen particle filter (give arguments how you conclude this)?

c.) Show the quality of your localization during the tests in task (2.1.b.) for different scenarios. Discuss the results and give ideas for potential reasons/problems. What are the limitations of the localization?

d.) Run the particle filter implementation on the real hardware race car. Provide the answers to the questions as in task (2.1.c.). It might be useful to record a ROS-bag to give you more flexibility without the need to do all your work on-site.

e.) Where there modifications (e.g. of parameters) needed to adjust your solution of task (2.1.a.) to the task (2.1.d.)? If yes, give details and possible reasons.

2.2 Improve the Localization

Sensor fusion is a method to improve the perception about the envirionment and/or the quality of the estimated state of a system. In our case the estimated state is the pose on the map. For proper performance of racing algorithmns the quality of this state estimation is crucial. We therefore want to combine all available sensor readings to improve it.

a.) Recap from task (2.1.b.) what data is processed by your chosen particle filter. Compare it to all available sensor ressources of our system (this might be different in simulation and on the real hardware car). What data might be valuable to improve the localization? How could this be incorporated?

b.) Improve the localization using one or more appropriate ROS-packages that help to incorporate measurements from sensors. The packages *robot_pose_ekf* [Sta22f] and *robot_localization* [Sta22e] might be interesing. But you are of course free to use any other package(s).

c.) Run the same tests and provide the same details as in task (2.1.b.) with the additional implementation from task (2.2.b).

d.) Run the additional implementation from task (2.2.b) on the real hardware race car. Provide the answers to the questions as in task (2.1.c.). Did the quality improve compared to the results from task (2.1.c.)? It might be useful to record a ROS-bag to give you more flexibility without the need to do all your work on-site.

e.) Where there modifications (e.g. of parameters) needed to adjust your solution of task (2.2.a.) to the task (2.1.d.)? If yes, give details and possible reasons.

Appendix

Grading

The following points can be achieved for each task of this exercise sheet:

Exercise	Points
1.1.a.	10
1.1.b.	6
1.1.c.	3
1.1.d.	3
1.1.e.	3
1.1.f.	9
1.1.g.	2
Subtotal	36
2.1.a.	10
2.1.b.	8
2.1.c.	4
2.1.d.	9
2.1.e.	3
Subtotal	34
2.2.a.	3
2.2.b.	10
2.2.c.	5
2.2.d.	9
2.2.e.	3
Subtotal	30
Grand Total	100

References

[Aut22] The Cartographer Authors. *Cartographer ROS Integration*. 2022. URL: https://google-cartographer-ros.readthedocs.io/en/latest/.

[Gio22] Wolfram Burgard Giorgio Grisetti Cyrill Stachniss. *GMapping*. 2022. URL: https://openslam-org.github.io/gmapping.html.

[Sta22a] Stanford Artificial Intelligence Laboratory et al. *AMCL*. 2022. URL: http://wiki.ros.org/amcl.

[Sta22b] Stanford Artificial Intelligence Laboratory et al. *gmapping*. 2022. URL: http://wiki.ros.org/gmapping.

[Sta22c] Stanford Artificial Intelligence Laboratory et al. *map_server*. 2022. URL: http://wiki.ros.org/map_server.

[Sta22d] Stanford Artificial Intelligence Laboratory et al. *Recording and playing back data*. 2022. URL: http://wiki.ros.org/ROS/Tutorials/Recording%20and%20playing%20back%20data.

[Sta22e] Stanford Artificial Intelligence Laboratory et al. *robot_localization*. 2022. URL: http://wiki.ros.org/robot_localization.

[Sta22f] Stanford Artificial Intelligence Laboratory et al. *robot_pose_ekf*. 2022. URL: http://wiki.ros.org/robot_pose_ekf.

[Wik22] Wikipedia contributors. *Simultaneous localization and mapping — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Simultaneous_localization_and_mapping&oldid=1081202080. [Online; accessed 8-April-2022]. 2022.

Acknowledgments

This course is based on F1TENTH Autonomous Racing which has been developed by the Safe Autonomous Systems Lab at the University of Pennsylvania (Dr. Rahul Mangharam) and was published under CC-NC-SA 4.0 license.