Machine Learning for Probabilistic Robotics with Webots

Master Thesis

Joan Sebastian Gerard Salinas

Promotor: Gianluca Bontempi

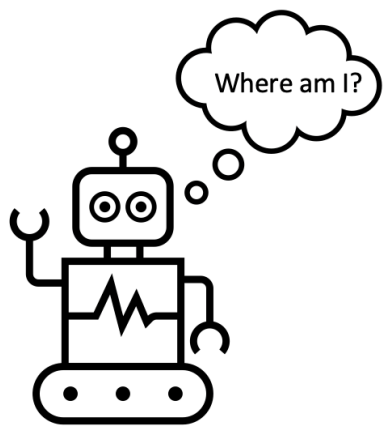
September 2020

Outline

Outline

Mobile Robotics

- A **mobile robot** is a vehicle that moves autonomously through navigation.
- **Navigation** is about controlling and operating the course of a robot. It is based on four blocks [1]:
  - *Sense:* the robot perceives its environment using its sensors.
  - *Localization:* the robot localize itself in the environment.
  - *Cognition:* the robot creates an execution plan.
  - *Control Action:* the robot executes its execution plan by modulating its output motors.
- This work focuses on the **localization** part.



Probabilistic Robotics for Robot Localization

Why is it difficult for a mobile robot to self-localize?

- There is no direct mapping from sensor measurements to a position in the environment.
- A GPS can be used but it does not offer the required precision, especially when the robot is in an indoor environment.
- The robot has to deal with **uncertainty** all the time. A deterministic approach requires extra features in the environment such as landmarks.

In order to deal with uncertainty and seeking to provide a reliable solution **probabilistic robotics** is used for localization: probability theory techniques applied to robotics.

Simulation for Robotics

- There are many robotics simulators but few of them are open-source and can be used in academia such as Webots and Gazebo.
- The use of simulation in robotics makes possible the comparison of different algorithms, reproducibility, and justification of the results [2].
- Here, **Webots** is used as a robotics simulator to perform the experiments.

Contribution

- An **architecture** illustrating how machine learning can be combined with probabilistic robotics algorithms to localize a robot in simulation.
- A **variant** of the **particle filter** algorithm is presented to localize the robot where the particles' weight is updated using predictive models.
- An **open-source plugin for Webots** simulator is implemented that illustrates the presented particle filter algorithm. This plugin can be used to show students the role of machine learning in robotics.

Motivation

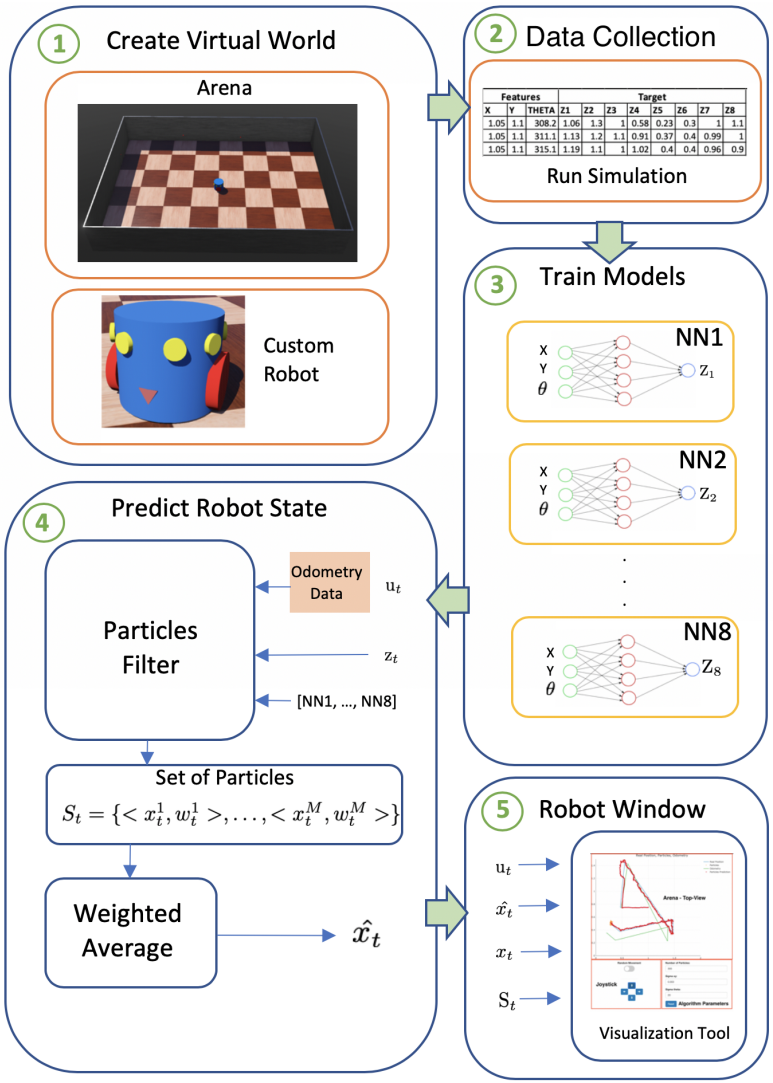
- There is a lot of research going on trying to improve mobile robot localization methods.
- The idea of using machine learning to empower probabilistic robotics is relatively new.
- The use of a robotics simulator allows us to perform many experiments within a custom environment and a custom robot.

Motivation: Large Scale Applications

- **Mamut** mobile robot automates data collection for farmers.
- **6 River Systems** and **Locus Robotics** supply mobile robots for the warehousing and order fulfillment business.
- **Alphabet, Amazon, FedEx,** and **UPS** are working on mobile robots for local package deliveries.
- Amazon will be using 120,000 robots by the end of this year, and the global market for mobile robots will surpass $3 billion, predicts Ash Sharma, research director at Interact Analysis[3].


Outline

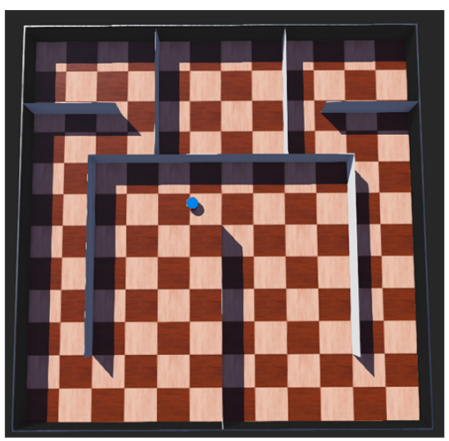
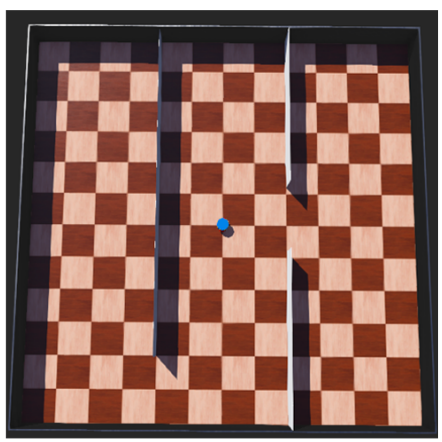
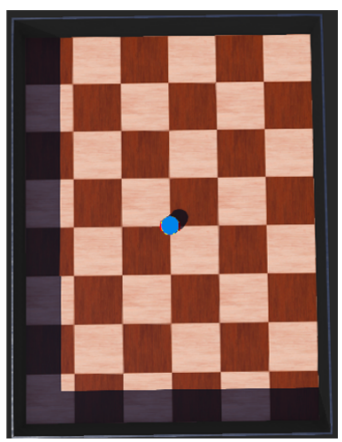
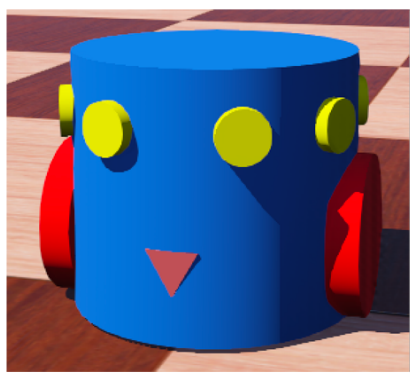
Architechture Overview



- Virtual world creation
- Data collection
- Prediction models training
- Robot state prediction
- Robot window

Methods

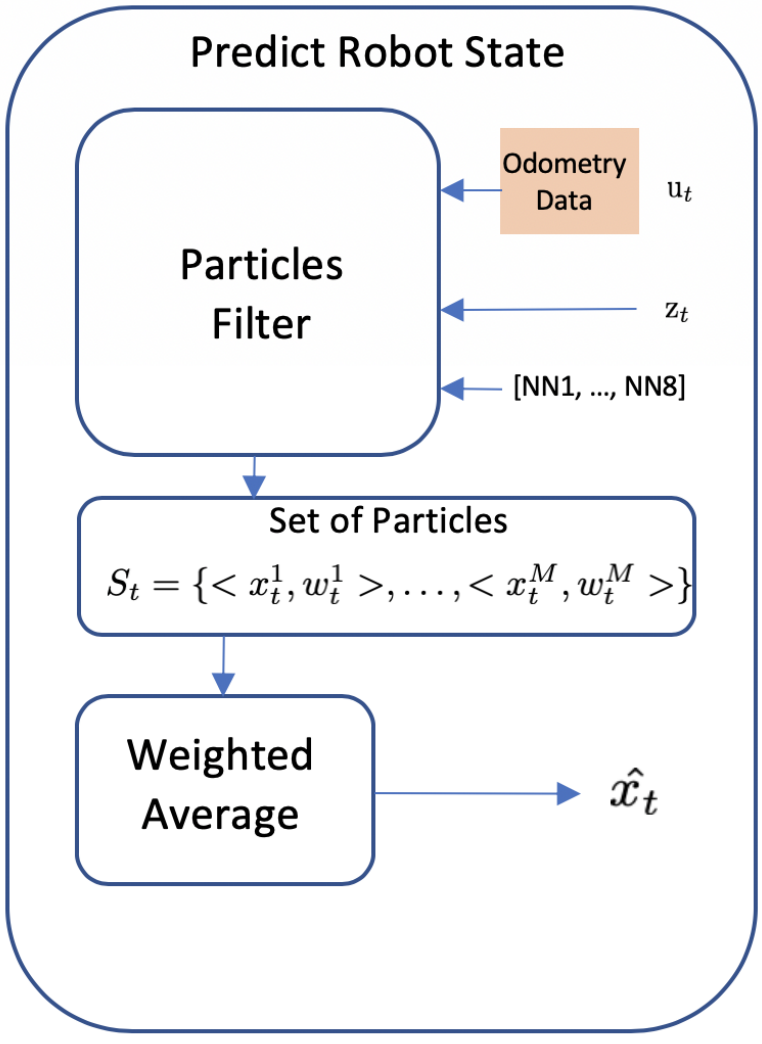
1. Virtual world creation
   - Creation of the virtual environment through the Webots interface.
   - Creation of a custom robot using the Webots interface.
   - The robot is configured to have eight laser distance sensors around the robot, two wheels, a compass sensor.
   - The laser sensors measure the distance between the sensor and the target object, in this case the walls.
2. Data collection
   - The robot is placed in the environment.
   - It moves randomly.
   - It is placed into another random location every 200 steps.
   - It collects sensor measurements together with robot states.

Methods

3. Train prediction models
   - Eight neural networks models are trained, assessed, and selected (one per sensor).
   - Their input is the robot state and their output is a sensor measurement.
   - The **learned function** is $\hat{z}_t = f(x_t, y_t, \theta_t)$ where (x_t, y_t, θ_t) is a robot state and $\hat{z}_t$ is the predicted sensor measurement.
4. Robot state prediction (localize the robot)
   - It uses the adapted particle filter to predict the robot state.
5. Robot window
   - A Webots plugin was created to visualize the particles state.
   - It shows the influence of the parameters on the algorithm.

Algorithm: Adapted Particle Filter



- Every particle m has a state $x_t^{[m]}$ and a weight $w_t^{[m]}$ associated with it.
- The particle filter output a set of particles. The weighted average is obtained to make a robot state prediction.

$$\hat{x}_t = \frac{1}{W_t} \sum_{1}^{m=M} w_t^{[m]} x_t^{[m]}$$

where W_t is the sum of all the particles' weight $W_t = \sum_1^{m=M} w_t^{[m]}$.

Algorithm: Particles Initialization

- For **local localization** the particles state are drawn from a normal distribution, where Θ, X, Y are random variables, (θ_0, x_0, y_0) is the initial robot state, and $(\sigma_\theta, \sigma_x, \sigma_y)$ are the standard deviation parameters for the robot rotation and translation:

$$\Theta \sim \mathcal{N}(\theta_0, \sigma_\theta)$$
$$X \sim \mathcal{N}(x_0, \sigma_x)$$
$$Y \sim \mathcal{N}(y_0, \sigma_y)$$

- For **global localization** the particles state are drawn from a uniform distribution, where env_x and env_y are the arena dimensions in the (x, y) coordinates respectively:

$$\Theta \sim \mathcal{U}(0, 360)$$
$$X \sim \mathcal{U}(0, env_x)$$
$$Y \sim \mathcal{U}(0, env_y)$$

Algorithm: Update Particles State

- Every time step the **particles state is updated** according to what the odometry data reports.
- Additionally, some noise is added from a normal distribution with mean 0 and standard deviation $\{\sigma_\theta, \sigma_x, \sigma_y\}$ parameters.
- Values are clipped according to the dimension of the environment.

Algorithm: Calculate Particles Weight

- Here is where the presented particle filter differs from other versions.
- For every particle a prediction of the sensor measurement is obtained $\hat{z}_t^i$ **using the learned function** $f(x_t, y_t, \theta_t)$. Here, i is the sensor index and **does not** represent "*to the power of*".
- This sensor measurement is compared to what the robot is perceiving to obtain an error term according to:

$$e_t^{[m]} = \sum_{i=1}^{i=8} |z_t^i - \hat{z}_t^i|^p$$

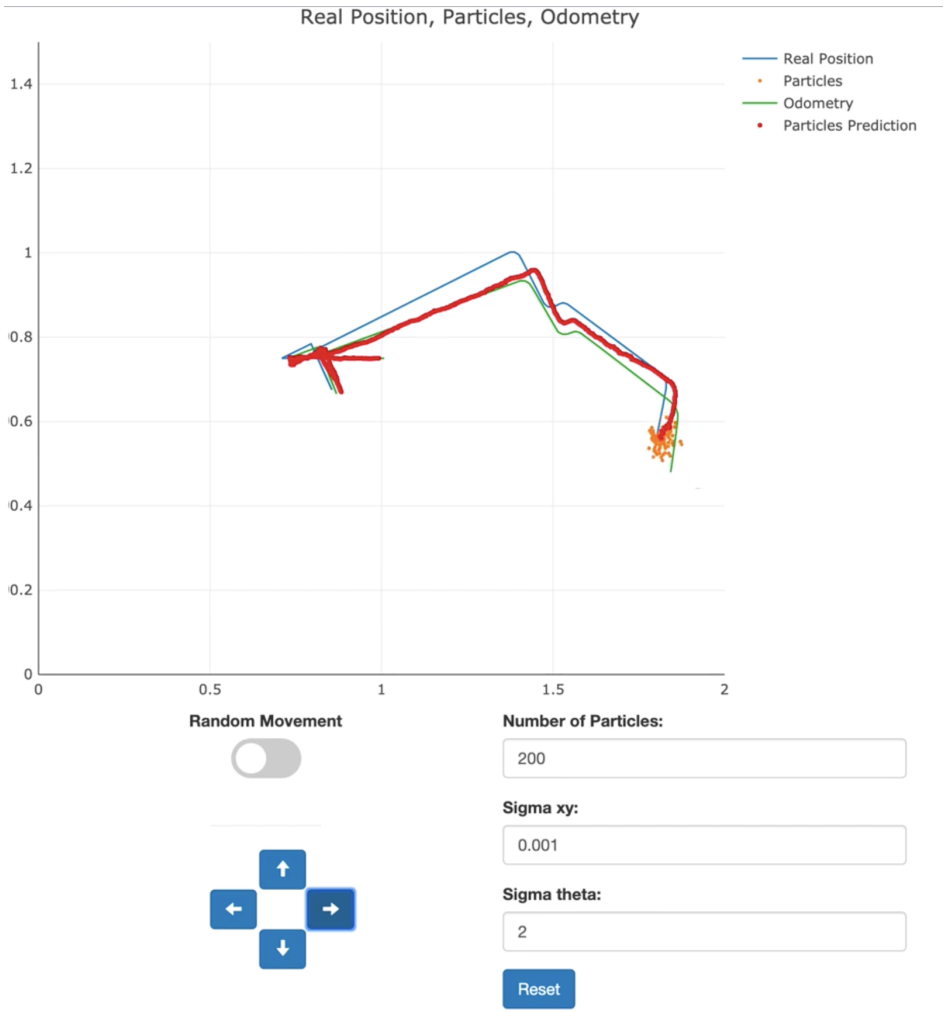
- The weight of the particle m is calculated as the inverse of the error term.

$$w_t^{[m]} = \frac{1}{e_t^{[m]}}$$

- **Resampling**: the particles are drawn randomly with replacement according to their weights.

Outline

Software Results



- The created Webots plugin serves mainly to **visualize** the particles state, the predicted robot state, the true robot state, and the odometry state estimation.
- It contains a window form that allows **setting** the particle filter **parameters**.
- It allows to **manually control** the robot movements or let the robot **move randomly**.

Outline

Predictive Model

- **1M** of samples were collected in simulation with sensor noise of 0.05.
- The data set contains the robot true position x, y, θ and the eight sensor measurements for that position.
- The data set was split in two, for **training** and **testing**.
- **Eight neural network models** were trained (one per sensor) using the collected data.
- The final selected model has **four hidden layers** with 3 (input), 16, 32, 64, 16, and 1 (output) neurons.

Particle Filter Accuracy Metrics

The cumulative error is defined as follows:

$$CTE_t = \sum_{1}^{i=t} \sqrt{(X_i - \hat{X}_i)^2 + (Y_i - \hat{Y}_i)^2}$$

$$CRE_t = \sum_{1}^{i=t} \sqrt{(\Theta_i - \hat{\Theta}_i)^2}$$

The root-mean-square deviation is defined as follows:

$$RMSDT_t = \sqrt{\frac{\sum_1^{i=t}(X_i - \hat{X}_i)^2 + (Y_i - \hat{Y}_i)^2}{t}}$$

$$RMSDR_t = \sqrt{\frac{\sum_1^{i=t}(\Theta_i - \hat{\Theta}_i)^2}{t}}$$

where (X_i, Y_i, Θ_i) is the true robot state and $(\hat{X}_i, \hat{Y}_i, \hat{\Theta}_i)$ is the predicted robot state by the proposed particle filter algorithm at time step i.

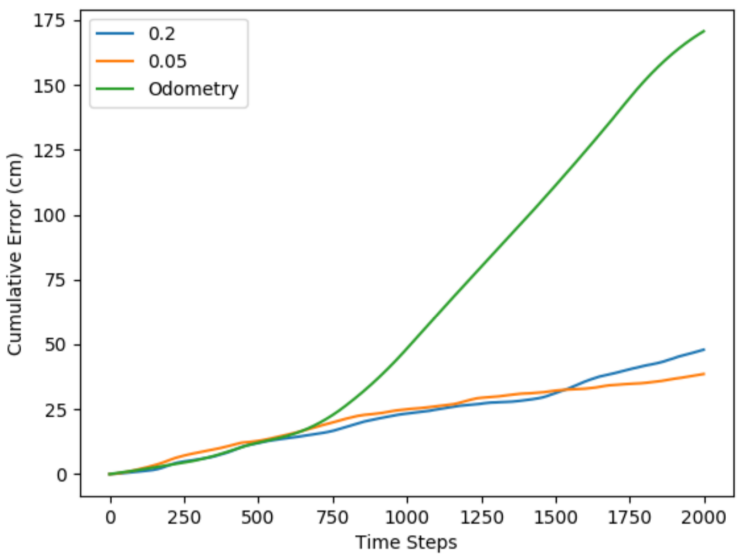
Experiment: Optimal Parameters

- The robot is placed in the **small arena** and it follows a static configured path.
- The same experiment is repeated with different values for the particle filter algorithm.
- Optimal values empirically found for particle filter:

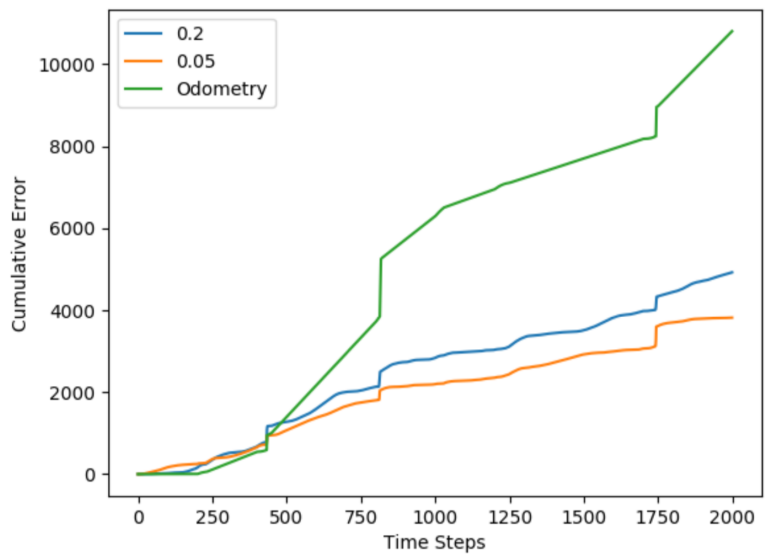
Parameter	**Value**
Number of particles m	1000
Standard deviation for particles translation σ_{xy}	0.0025
Standard deviation for particles rotation σ_{θ}	0.5
Hyperparameter p	1

Experiment: Noisy Sensors

- The robot **sensors noise** is configured to have a standard deviation of **0.2**
- Data collection is performed in simulation obtaining **1M samples**.
- The previously selected model is retrained with the recently collected data set.
- The robot is placed in the **small arena** and it follows a static configured path.



(a) CTE



(b) CRE

Experiment: Number of Sensors

- Only three sensors are used instead of eight.

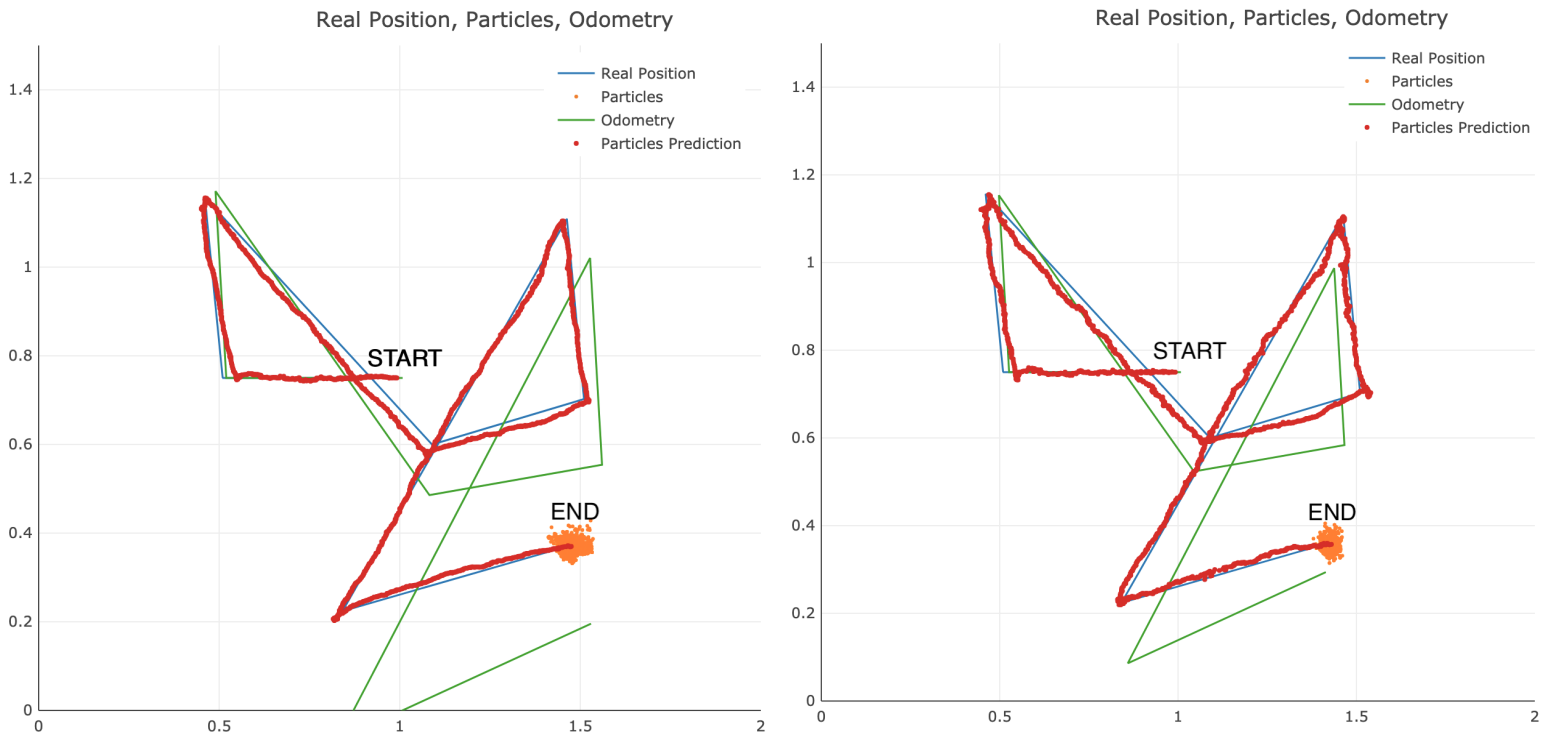
# Sensors	RMSDT (cm)	RMSDR (degrees)
3	6.83	11.48
8	2.31	9.32
Odometry	9.68	21.48

Table: RMSD metrics for 3 sensors, 8 sensors, and odometry.

Experiment: Incremental Wheel Diameter

- The wheel *grows*. In other words, the wheel diameter is incremented at each time step, as follows:

$$D_l \leftarrow D_l + 0.00075$$
$$D_r \leftarrow D_r + 0.00075$$



Experiment: Multiple Environments

- The robot is placed in a **simple, medium,** and **medium-high** complexity environments.
- The robot follows a static configured path.
- The data collection and model training is made for each environment.

Arena Complexity	**RMSDT**	**RMSDT (odometry)**	**RMSDR**	**RMSDR (odometry)**
Simple	2.64	9.68	8.68	21.48
Medium	5.24	22.41	134.59	146.73
Medium-High	8.93	27.39	113.28	118.00

Experiment: Global Uncertainty

- The experiment is repeated **ten times**. Each time the robot is placed in a new different location over the arena.
- The initial robot state is **not known** by the particles.
- The particles succeed to find the true robot state in **seven** out of ten tries.

Outline

Conclusion

- The presented particle filter is **effective** to deal with both **local** and **global** localization.
- The use of **neural networks** as predictive models is **fast** when using a significant amount of particles.
- The developed **plugin** was useful to **visualize** the predicted robot states and to understand the **influence** of the algorithm **parameters**.
- The **didactic role** of the developed plugin is that it can be used to show **students** the **role of machine learning** in robotics.

Future Work

- Make use of an **embedded map** of the environment and predict the particles likelihood instead of sensor measurements.
- Moving to **visual localization** and **differentiable programming**.
- Use a robot with an integrated **camera** instead of laser sensors.
- Collect synthetic data using a **hybrid approach**.
- Add semantic information to the maps such as **labeling the objects**.
- As a long-term goal, implement navigation techniques that are guided by **voice commands**.

References


R. Siegwart et al. *Introduction to autonomous mobile robots.* 2011, pp. 265–366.


Francesco Amigoni and Viola Schiaffonati. "Good Experimental Methodologies and Simulation in Autonomous Mobile Robotics". In: vol. 314. Sept. 2010, pp. 315–332. DOI: 10.1007/978-3-642-15223-8_18.


Keith Shaw. *The Robotics Sector in 2020 and Beyond: Predictions from Industry Gurus*. 2019. URL: https://www.roboticsbusinessreview.com/news/the-robotics-sector-in-2020-and-beyond-predictions-from-industry-gurus/ (visited on 08/30/2020).

Outline

Thank you for your attention!
Do you have any questions?