Algorithms for Autonomous Navigation of Mobile Robots Near Obstacles

Research Report

Roman Eidelman

Supervisor: Professor Alexey S. Matveev

Sirius University

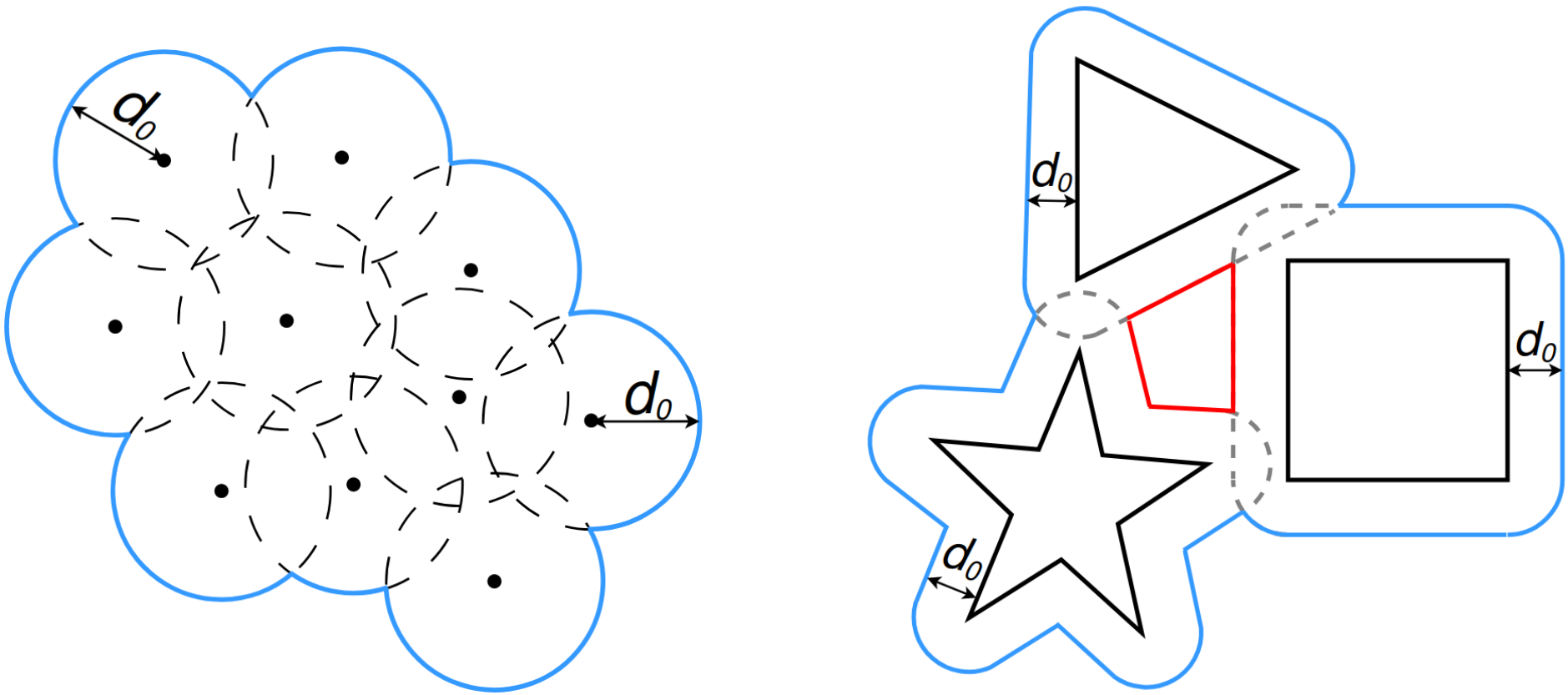
December 24, 2024

Content

Problem Statement

- An underactuated nonholonomic **Dubins-car** robot with a lower-limited turning radius travels with a constant speed in a plane, which hosts unknown complex objects. In its local frame, the robot has sensorial access only to the part of the scene that is within a finite zone of visibility and in direct line of sight.
- It is required to approach and then circumnavigate the objects, with maintaining a given distance to the currently nearest of them, so that the ideal targeted path is the **equidistant** curve of the set of the objects. The focus is on the case where this curve cannot be perfectly traced due to excessive contortions and singularities. So the objective is restated as that of automatically finding, approaching and repeatedly tracing an approximation of the equidistant curve that is the best among those trackable by the robot with respecting safety concerns.
- Reference paper DOI: https://doi.org/10.1016/j.robot.2024.104649

Equidistant



(a) (b)

Figure: Examples of **equidistant**

Dubins-car model

- Dubins-car model robot is a non-holonomic under-actuated planar robot, which travels with a constant speed $v > 0$ and is driven by the angular velocity ω limited in absolute value by a constant $\bar{\omega} > 0$:

$$\begin{cases} \dot{\boldsymbol{r}} = v\vec{e}(\theta), \vec{e}(\theta) := \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix}^\top \\ \dot{\theta} = \omega \in [-\bar{\omega}, \bar{\omega}] \end{cases} . \tag{1}$$

- Here $\boldsymbol{r}$ and θ give the robot's location and orientation, respectively; Eqs. (1) capture the robot's capacity to move over paths whose curvature radius $\geq R_{min} = \frac{v}{\bar{\omega}}$.

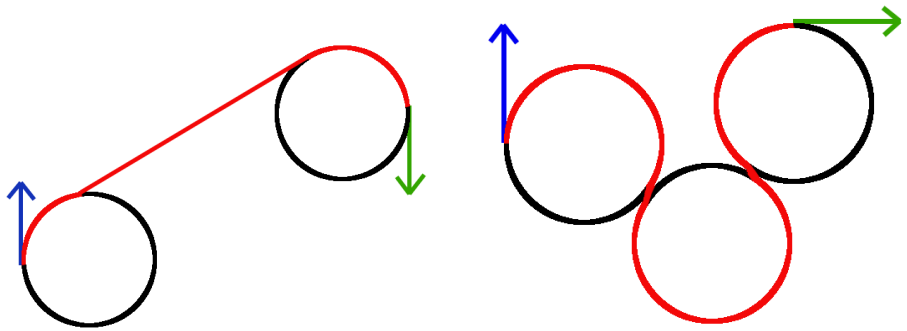


Figure: Examples of Dubins-car paths

Algorithm

- The control ω is calculated according to the following navigation rule:

$$\omega(t) = \bar{\omega} \cdot \textbf{\textit{sgn}}(\dot{d}_R(t) + \mu\chi[d_R(t)]), \tag{2}$$

where $\mu > 0$ is a tunable parameter, χ is a continuous and piece-wise smooth function $(\chi'(z) > 0, \forall z; \chi(0) = 0)$.

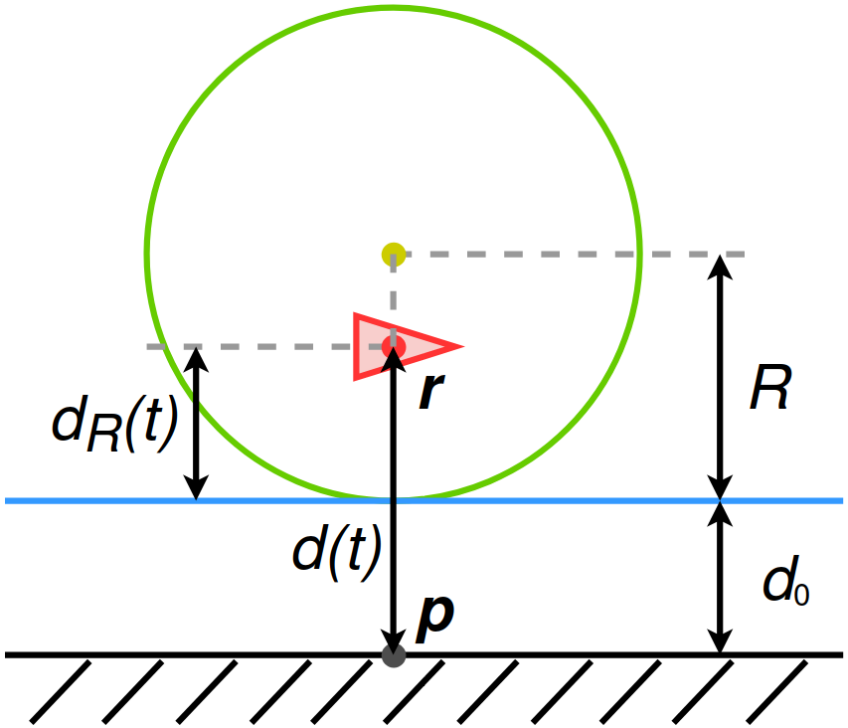
- The $d_R(t)$ and $\dot{d}_R(t)$ are calculated by following equations:

$$d_R(t) := \begin{cases} d(t) - d_0 & \text{in contact mode C} \\ R - ||\boldsymbol{r}(t) - \boldsymbol{v}[A(t)]|| & \text{in gap mode G} \end{cases}, \tag{3}$$

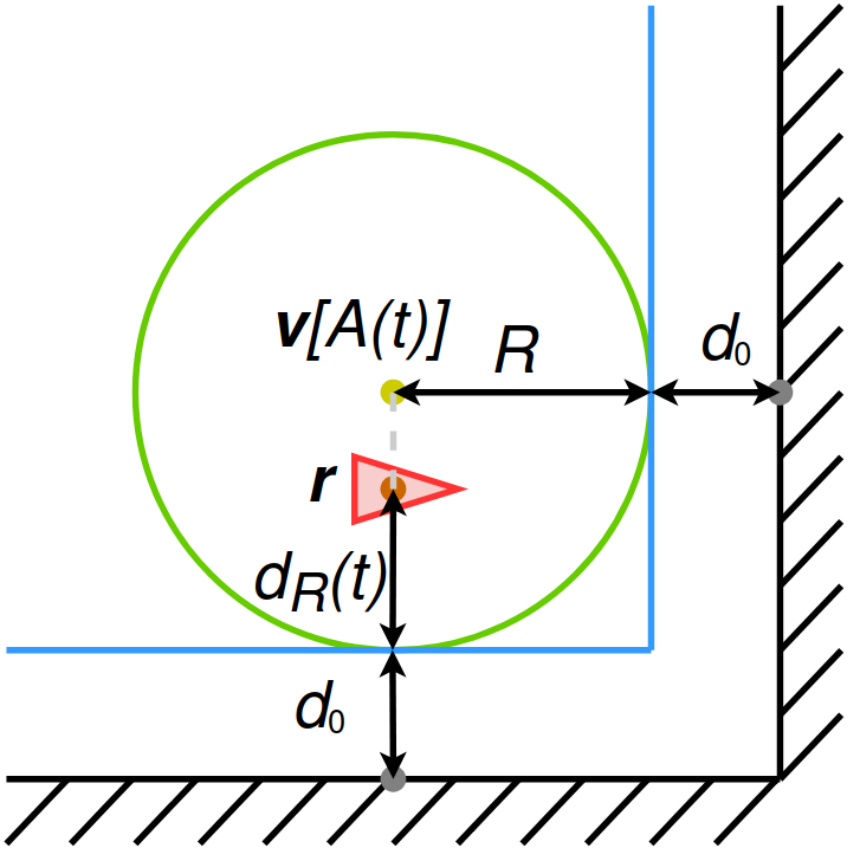
$$\dot{d}_R(t) := v \cdot \begin{cases} N[\boldsymbol{r}(t), \boldsymbol{p}(t)] \cdot \vec{e} & \text{in contact mode C} \\ N[\boldsymbol{v}[A(t)], \boldsymbol{r}(t)] \cdot \vec{e} & \text{in gap mode G} \end{cases}, \tag{4}$$

where $d(t)$, the current distance from the robot to the obstacle; d_0, the targeted distance; R, turning radius; $\boldsymbol{v}[A(t)]$, center of a disk in mode G; $\boldsymbol{p}(t)$, closest point of the obstacle; $N[a, b]$, normalized to the unit length vector $b - a$.

Modes



(a) Contact mode C



(b) Gap mode G

Figure: Algorithm modes

Simulation

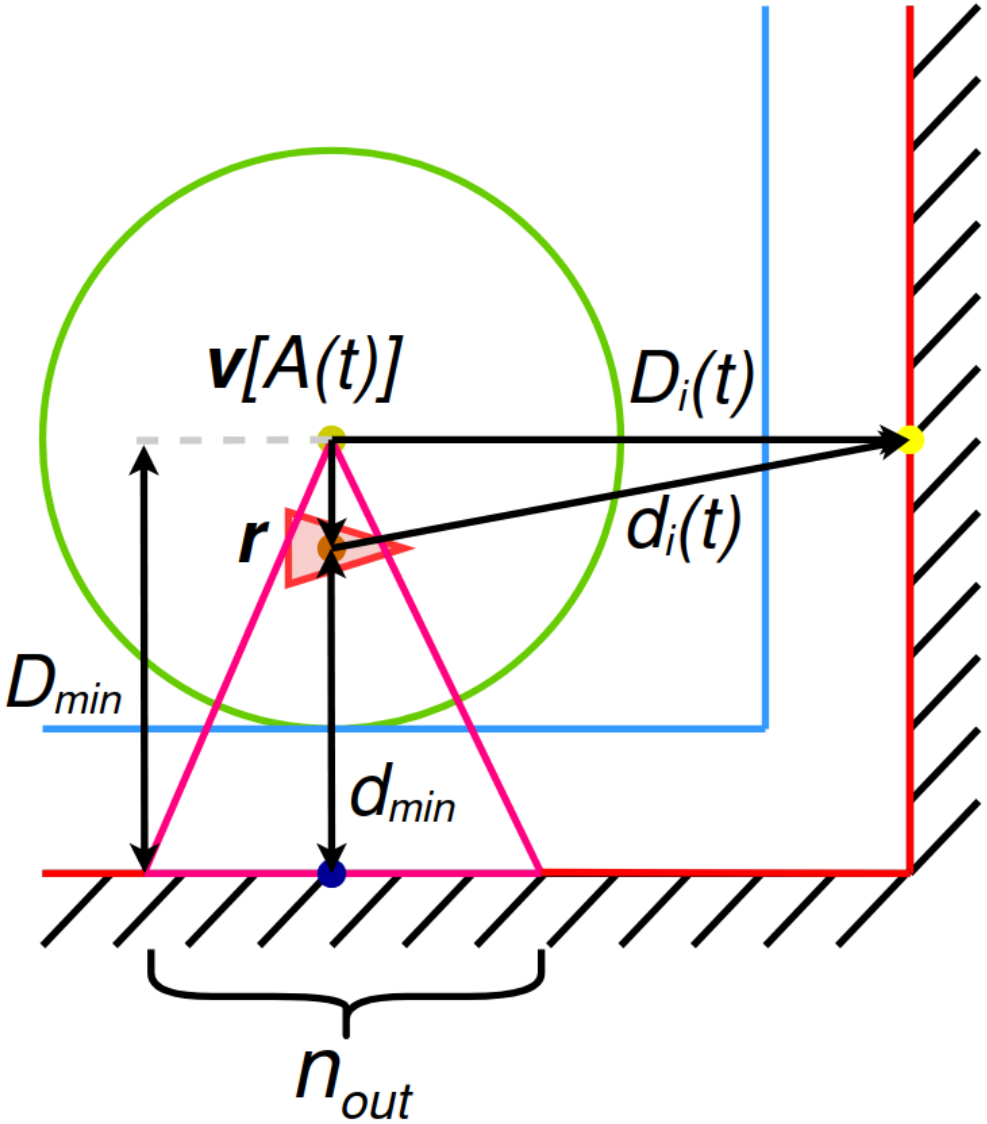


Figure: Calculating

Simulation

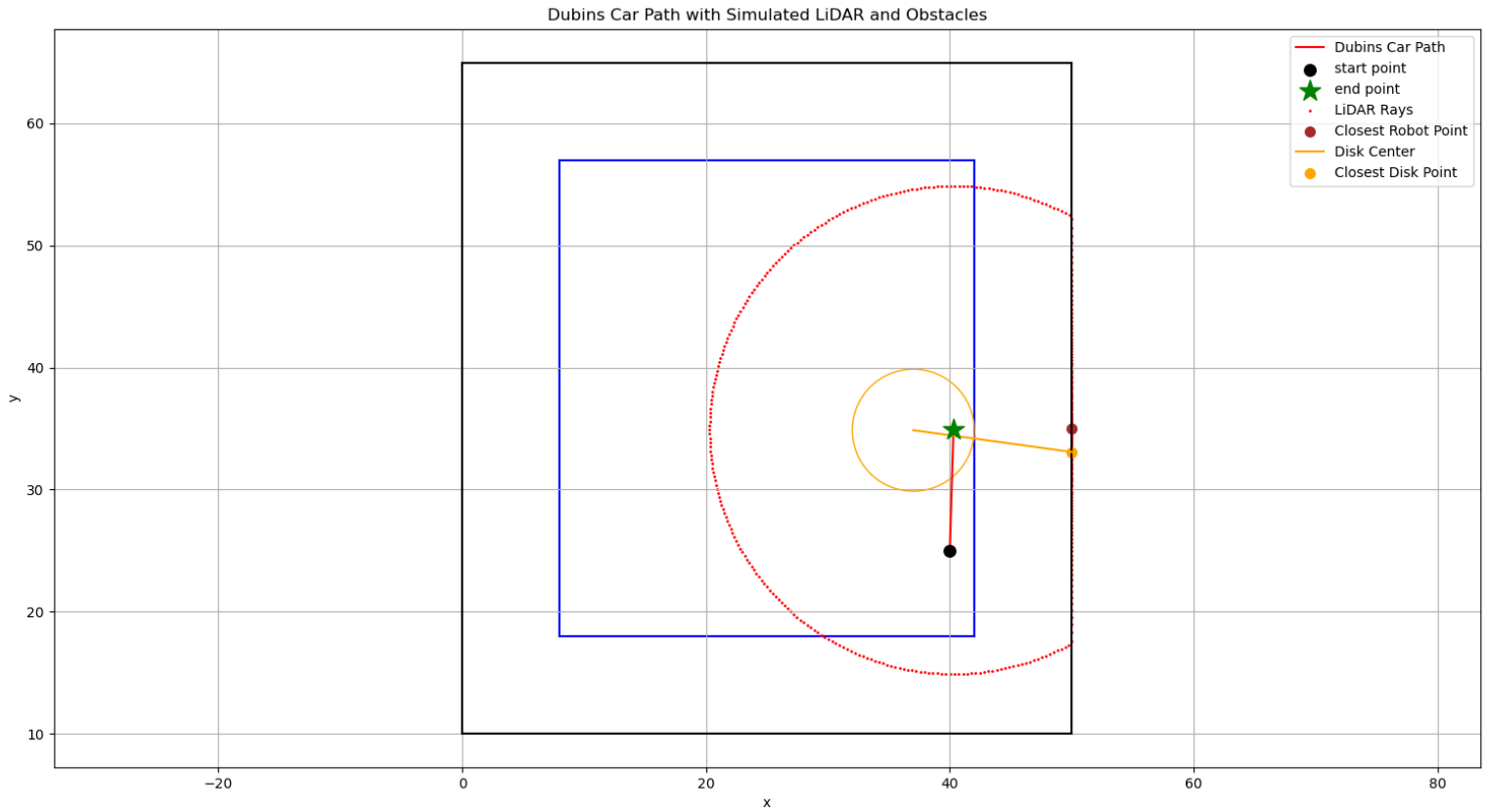
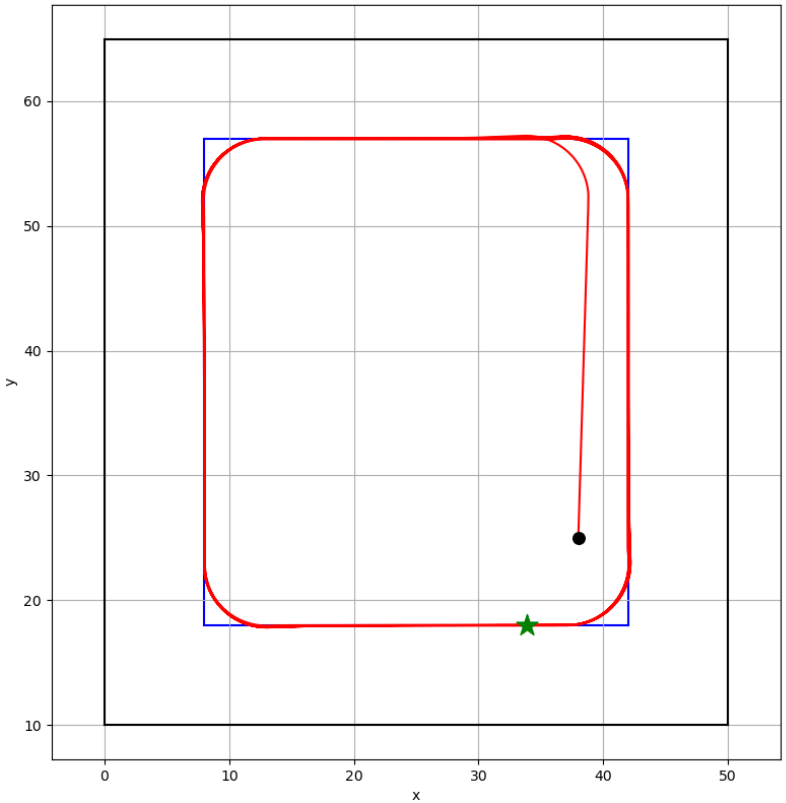
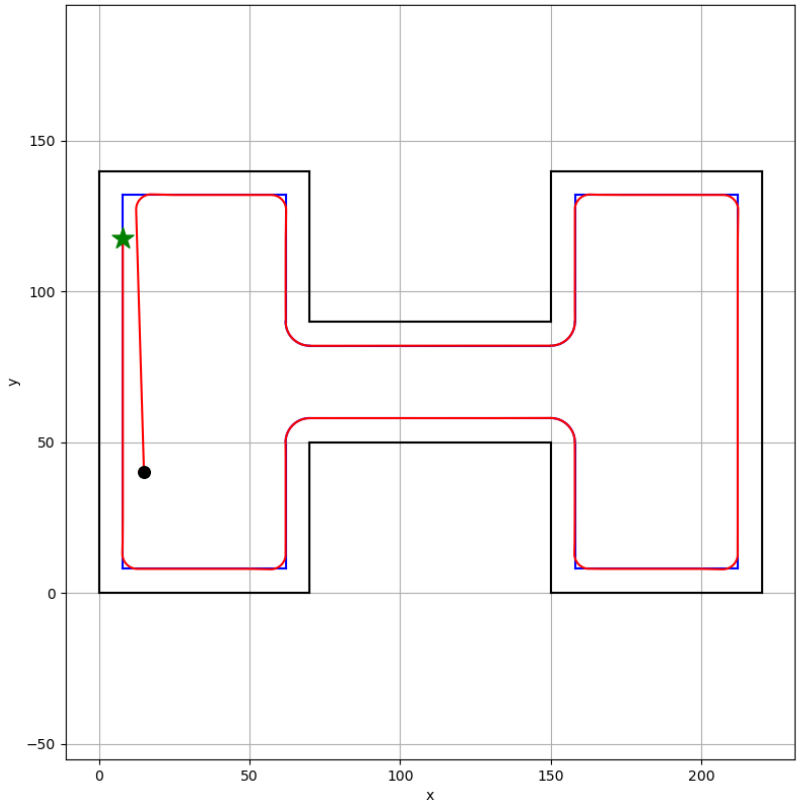


Figure: Simulation in Python 3

Testing of the Navigation Law



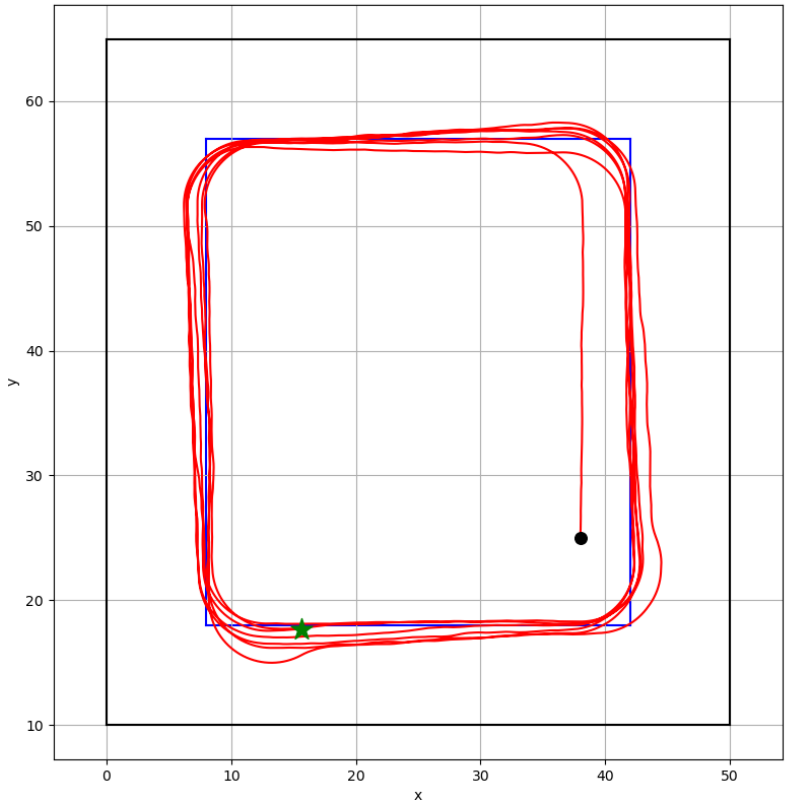
(a) Obstacle 1



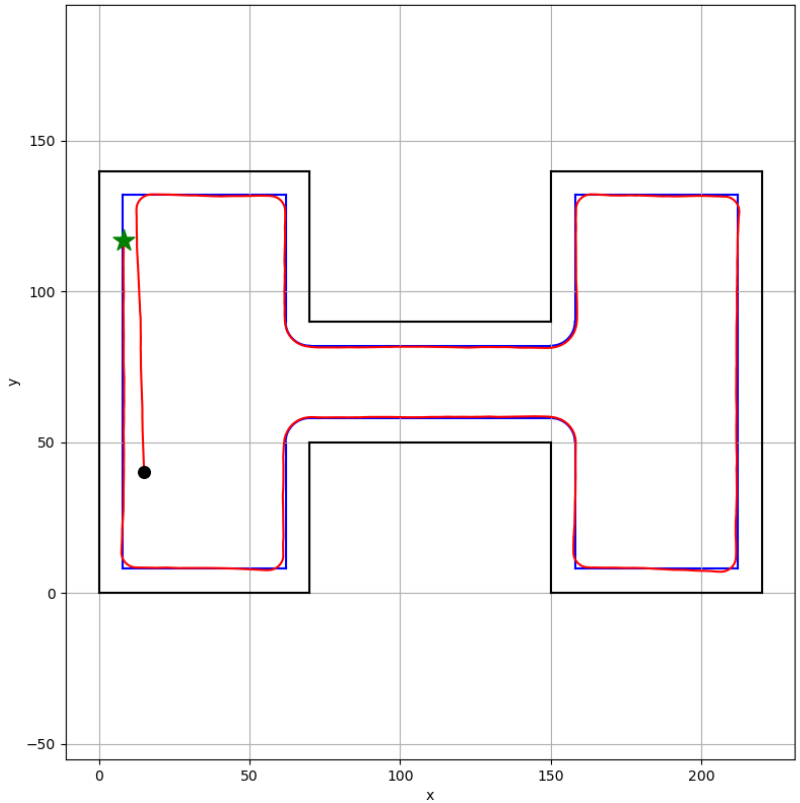
(b) Obstacle 2

Figure: Simulation without sensor noise

Experiments with noise



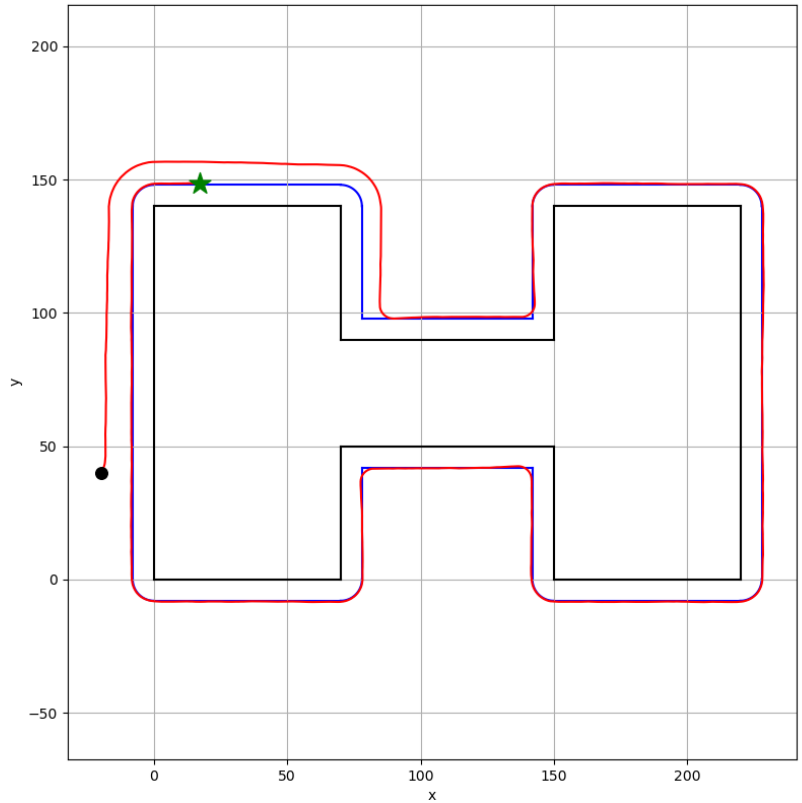
(a) Obstacle 1: v=0.1;



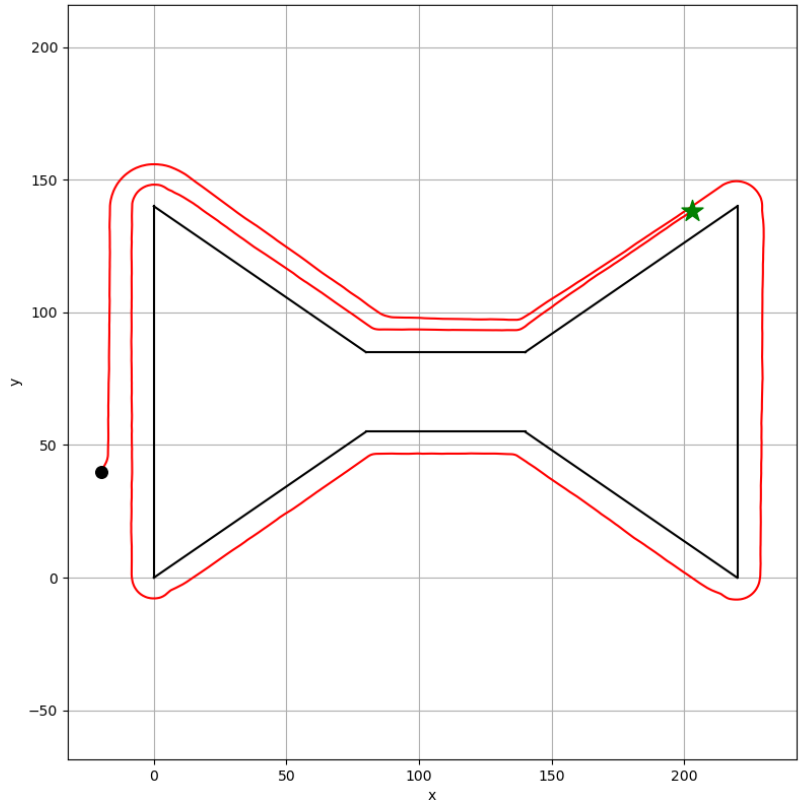
(b) Obstacle 2

Figure: 1 - Experiments with noise

Experiments with noise



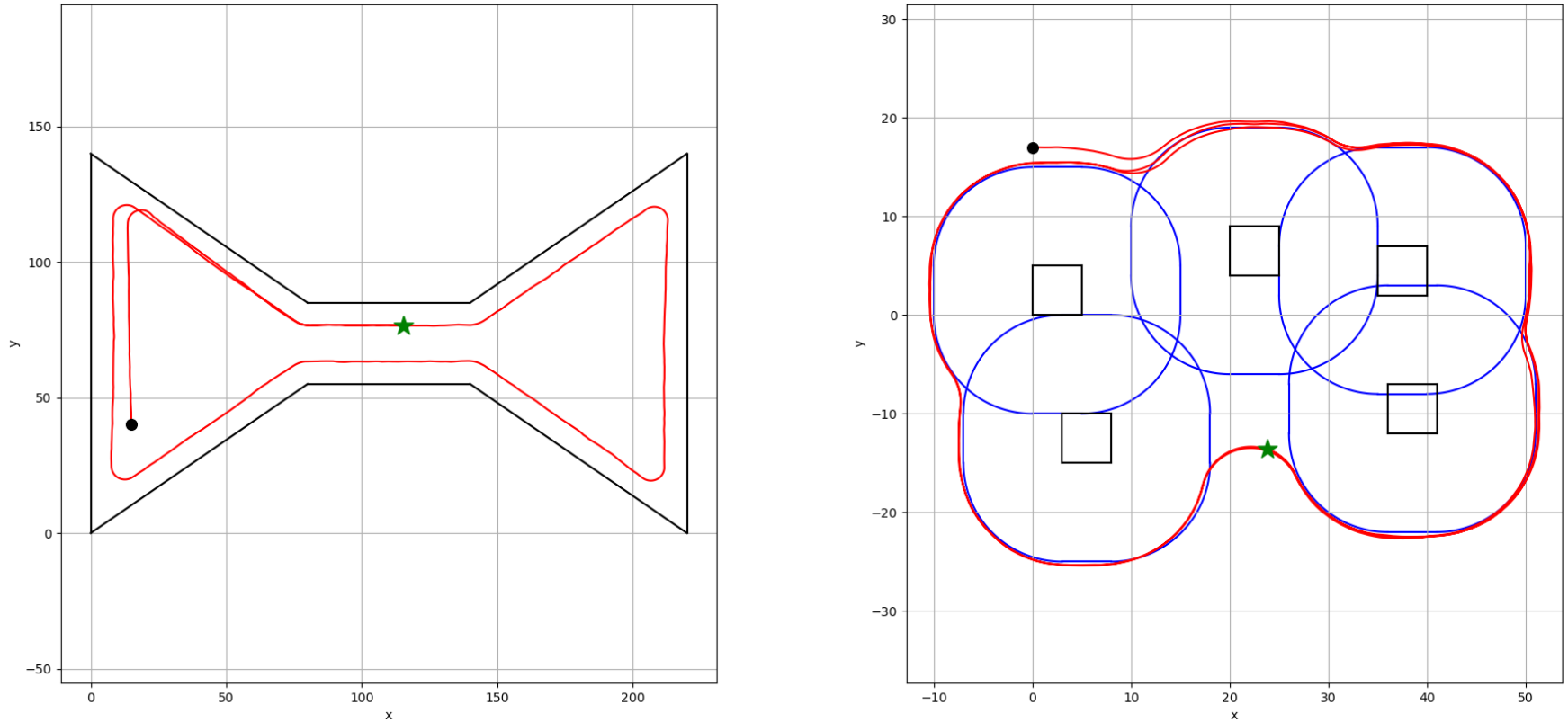
(a) Obstacle 3



(b) Obstacle 4

Figure: 2 - Experiments with noise

Experiments with noise

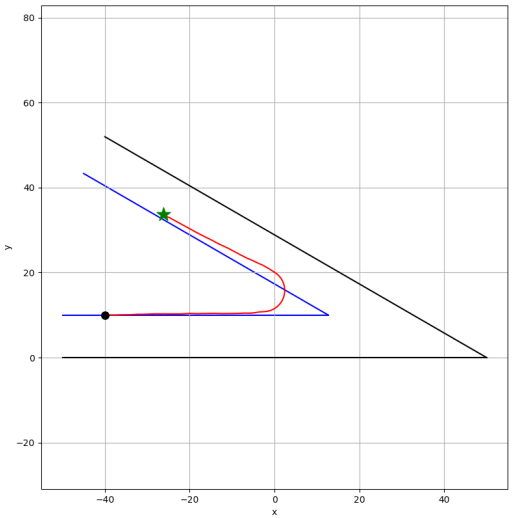


(a) Obstacle 5

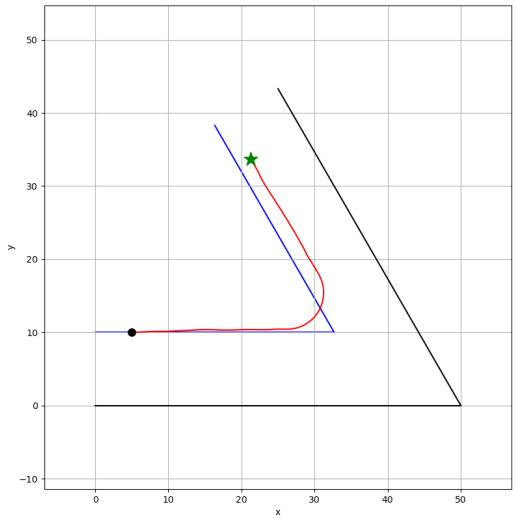
(b) Obstacle 6

Figure: 3 - Experiments with noise

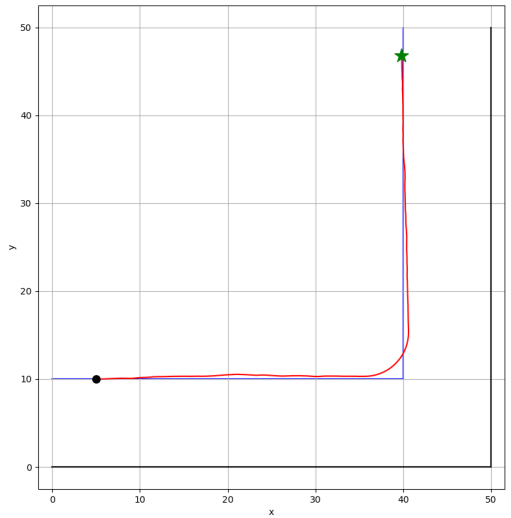
Problems



(a) Corner $\frac{\pi}{6}$

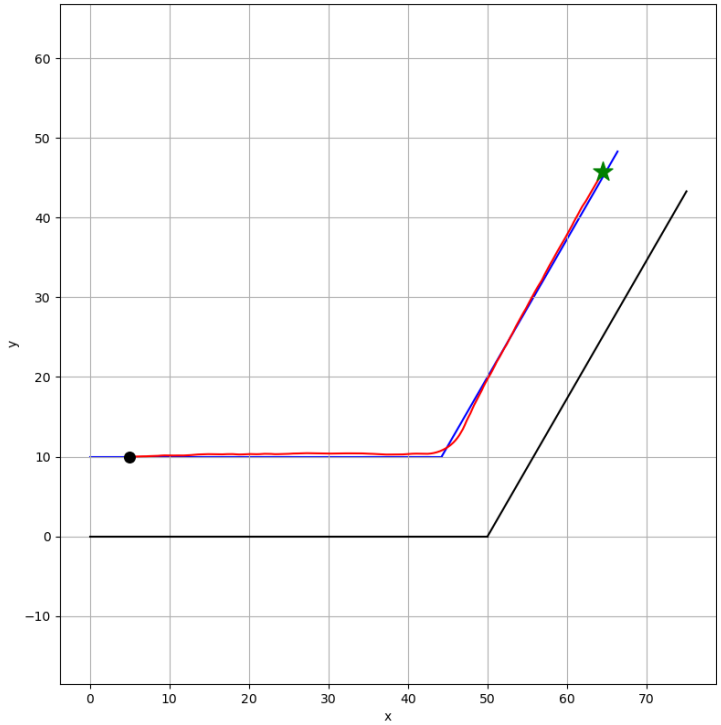


(b) Corner $\frac{\pi}{3}$

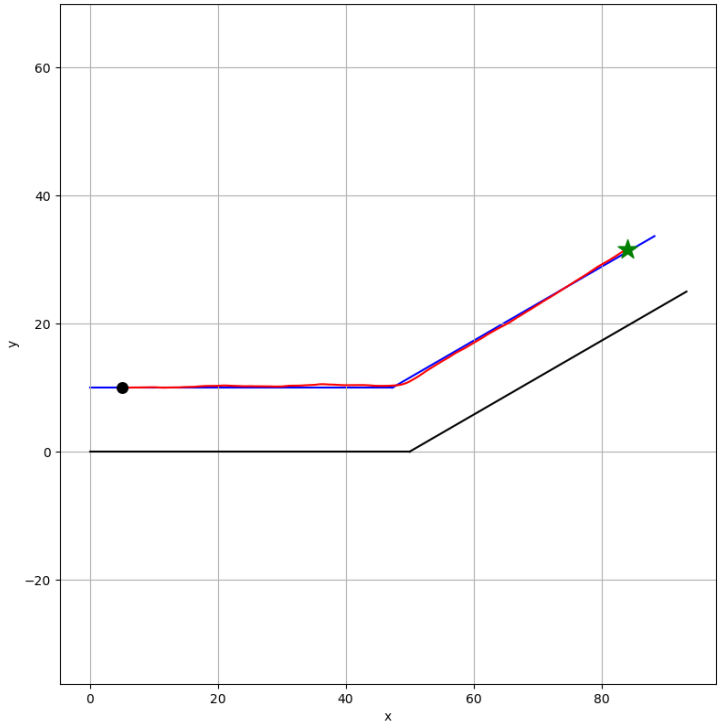


(c) Corner $\frac{\pi}{2}$

Problems

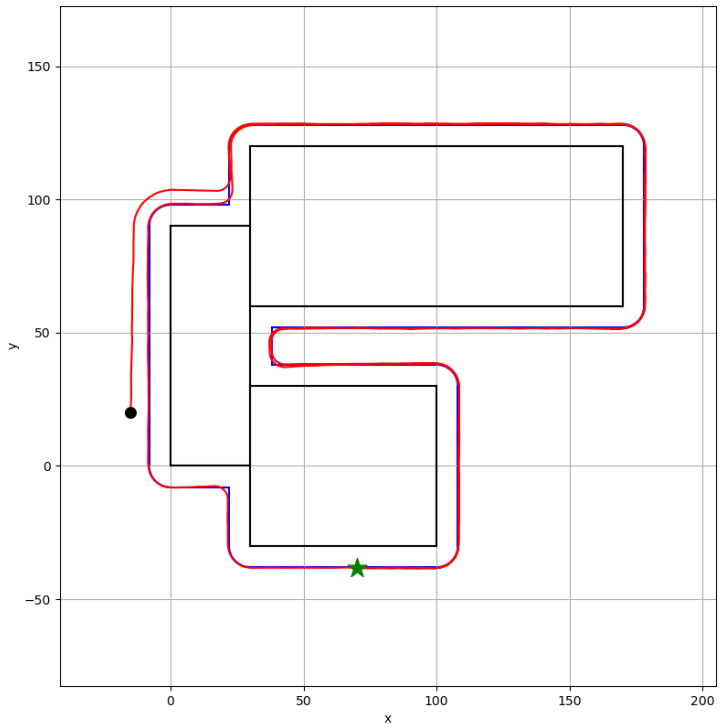


(a) Corner $\frac{2\pi}{3}$

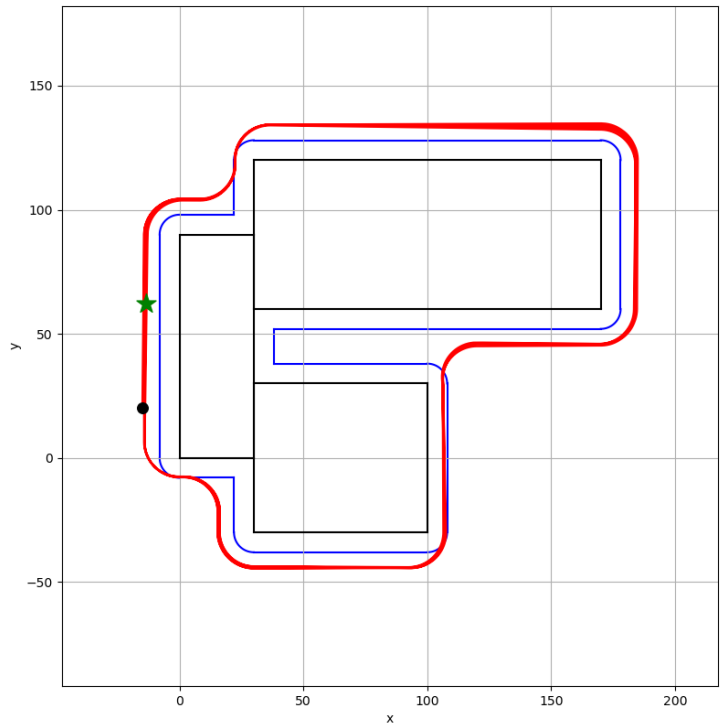


(b) Corner $\frac{5\pi}{6}$

Changing Linear Velocity



(a) $v = 0.1;\ R_{min} = 5$



(b) $v = 0.28;\ R_{min} = 14$

Conclusions

- The algorithm for autonomous navigation of a mobile robot near obstacles using sliding mode was modeled.
- Experiments were conducted with various types of obstacles
- Difficulties were identified in the algorithm's performance under noise conditions
- The next tasks are:
  - Transferring the simulation to Gazebo (ROS2)
  - Developing a regulator for control of linear velocity
  - Performing experiments on real robot