AC555
Obstacle Avoidance

Elena Pîndichi
Simon Abrial
Sinda Hichri

Université Grenoble Alpes Institut Nationale Polytechnique Ecole Supérieure en Systèmes Avancés et Réseaux

Wednesday, 15 January 2025

Table of Contents

Table of Contents

Idea

Improving the Obstacle Avoidance Algorithm:

- Generation of ellipses around the obstacles.
- Creation of moving obstacles by adding constant velocities.
- Implement the MPC controller on the TurtleBot and solving the optimziation problem.

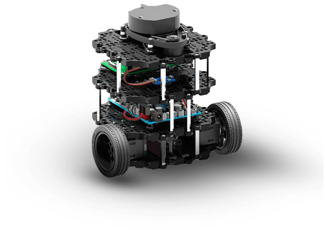


Table of Contents

Obstacles

We defined each obstacle's vertices and created them using an existing library in Python, named *Polygon*.

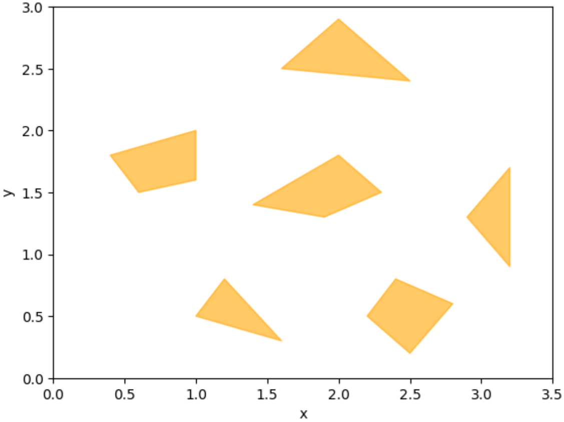
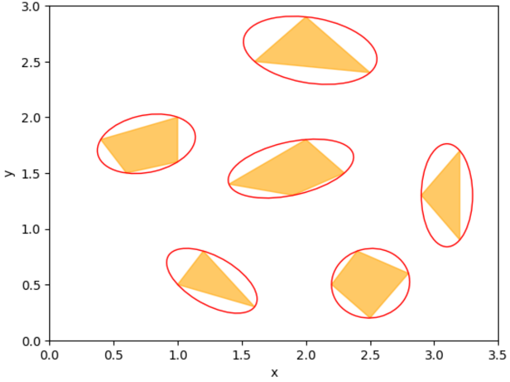


Figure: Polygons

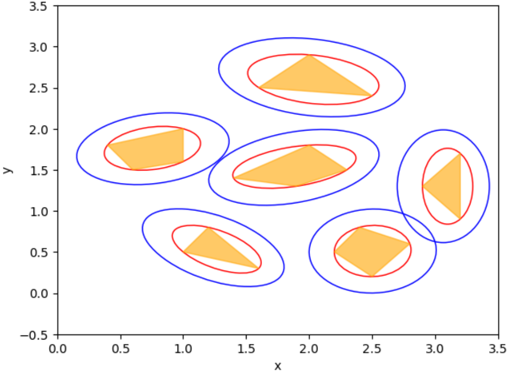
Ellipses

The Minimum Volume Enclosing Ellipsoid (MVEE) algorithm is used to find the smallest ellipsoid that encloses a given set of points in n-dimensional space. This ellipsoid is expressed as:

$$\mathcal{E} = \left\{ \mathbf{x} \in \mathbb{R}^n \mid (\mathbf{x} - \mathbf{c})^T \mathbf{A} (\mathbf{x} - \mathbf{c}) \leq 1 \right\}$$



(a) First Ellipses



(b) Enlarged Ellipses

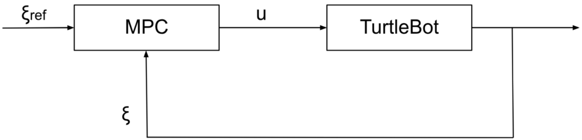
Table of Contents

Mathematical Model

The dynamics of the robot with respect to the control input vectors are:

$$\zeta = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad \dot{\zeta} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ \omega \end{bmatrix} \qquad u = \begin{bmatrix} V \\ \omega \end{bmatrix}$$

The controller receives data concerning the current system's state and, based on a target point, set by the operator, computes the optimal control inputs.



Dynamics of the system and Cost

The dynamics are nonlinear and the *sampling period* is used as the discrete system provides the state at equally divided samples of time ($T_s = 0.1$):

$$\begin{aligned} x(k+1) &= x(k) + T_s * V(k)\cos\theta(k) \\ y(k+1) &= y(k) + T_s * V(k)\sin\theta(k) \\ \theta(k+1) &= \theta(k) + T_s * \omega(k) \end{aligned}$$

The predictive control feedback law is computed by minimizing a predicted performance cost, which is defined in terms of the predicted sequences $\mathbf{u}$, $\mathbf{x}$. The predicted cost has the general form:

$$J(x_k, \mathbf{u}_k) = x_N^\top P x_N + \sum_{i=0}^{N-1} \left(||x_{i|k} - x_{ref}||_Q^2 + ||u_{i|k}||_R^2 \right)$$

Optimization cost function for static obstacles

The optimal control sequence for the problem of minimizing the predicted cost is denoted $\mathbf{u}_N^*(x_k)$ and we can rewrite the optimization cost function as:

$$\mathbf{u}_N^\star = \arg\min_{u_N} \left(x_N^\top P x_N + \sum_{k=0}^{N-1} \left((x_k - x_{ref})^\top Q (x_k - x_{ref}) + u_k^\top R u_k \right) \right)$$

$$\begin{aligned} \text{s.t.} \qquad x_{k+1} &= x_k + T_s * f(x_k, u_k), \\ x_{min} \leq x_{k+1} - x_k &\leq x_{max}, \\ u_{min} \leq u_k &\leq u_{max}, \\ (\mathbf{x_{k+1}} - \mathbf{c_{obs_i}})^T \mathbf{A_{obs_i}} (\mathbf{x_{k+1}} - \mathbf{c_{obs_i}}) &> 1, \\ \mathbf{A_{obs_i}} \in \mathbb{R}^{n \times n}, \mathbf{c_{obs_i}} \in \mathbb{R}^n, i &\in N_{obstacles}. \end{aligned}$$

Optimization cost function for moving obstacles

The optimal control sequence for the problem of minimizing the predicted cost while the obstacles are moving is denoted $\mathbf{u}_N^*(x_k)$ and we can rewrite the optimization cost function as:

$$\mathbf{u}_N^\star = \arg\min_{u_N} \left(x_N^\top P x_N + \sum_{k=0}^{N-1} \left((x_k - x_{ref})^\top Q (x_k - x_{ref}) + u_k^\top R u_k \right) \right)$$

$$\begin{aligned}
\text{s.t.} \qquad x_{k+1} &= x_k + T_s * f(x_k, u_k), \\
x_{min} \leq x_{k+1} - x_k &\leq x_{max}, \\
u_{min} \leq u_k &\leq u_{max}, \\
\left(\mathbf{x_{k+1}} - \mathbf{c_{obs_{i|k}}}\right)^T \mathbf{A_{obs_{i|k}}} \left(\mathbf{x_{k+1}} - \mathbf{c_{obs_{i|k}}}\right) &> 1, \\
\mathbf{A_{obs_{i|k}}} \in \mathbb{R}^{n\times n}, \mathbf{c_{obs_{i|k}}} \in \mathbb{R}^n, i &\in N_{obstacles}.
\end{aligned}$$

Weight Matrices

The weight matrices that we have chosen after multiple tests are:

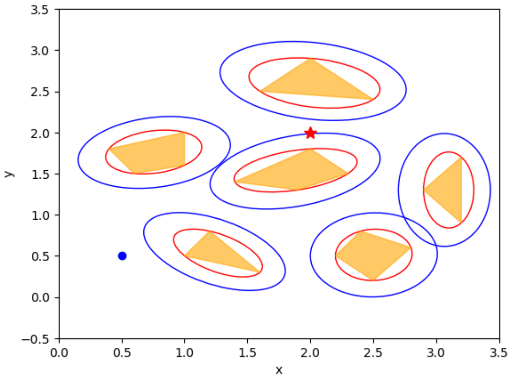
$$Q = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix} \qquad R = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \qquad P = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix},$$

with $N_{pred} = 100$.

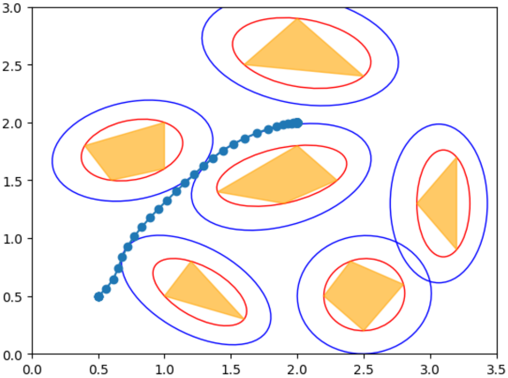
Computation time

The computation time for solving the problem is 10 seconds.

Results



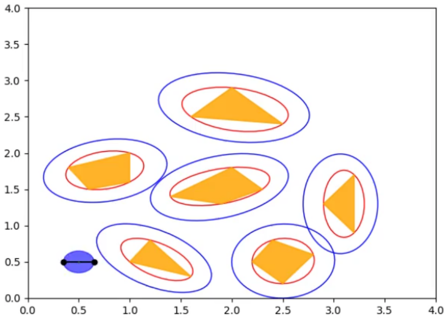
(a) Initial and Goal points



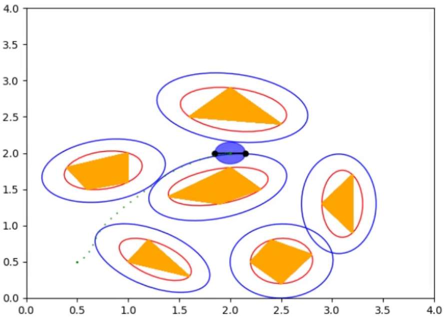
(b) Computed Trajectory

Figure: Static Obstacles

Results



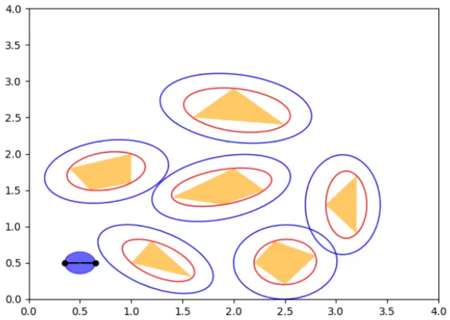
(a) Initial position



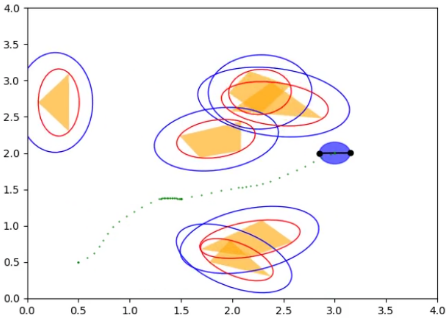
(b) Final Position

Figure: Static Obstacles

Results



(a) Initial position



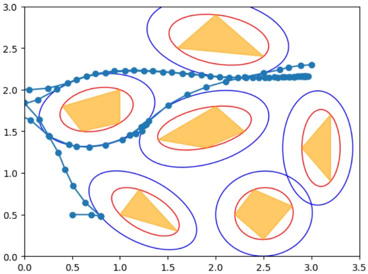
(b) Final Position

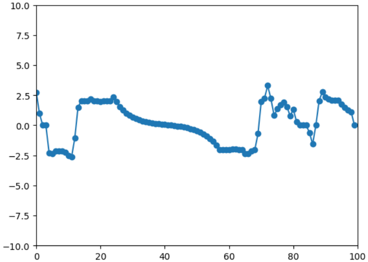
Figure: Dynamic Obstacles

Table of Contents

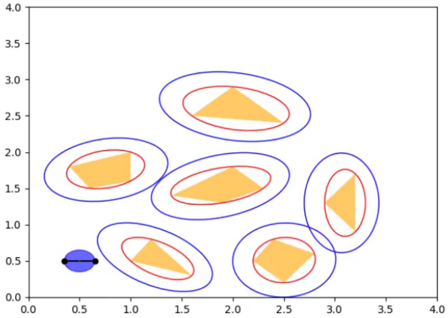
Perturbations

- Addition of noisy perturbations on the first state of x_k.
- ξ_{ref}
- Relaxing the input constraints.
- Random perturbations between [-0.1, 0.1].

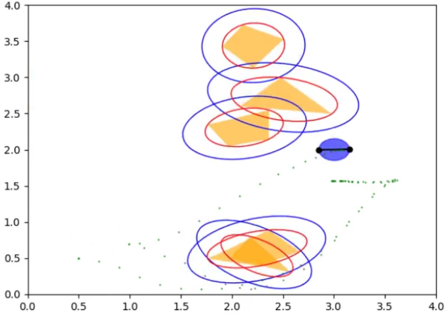




Results



(a) Initial position



(b) Final Position

Figure: Dynamic Obstacles

Table of Contents

Conclusion and Future Directions

Future Directions:

- Implementation of two robots that go to the same goal point.
- Making one robot follow another one while avoiding obstacles.

Conclusions:

- Implementation of two MPC problems.
- Creating an obstacle avoidance algorithm for moving objects.
- Understanding of how random perturbations might affect the behavior of the robot.

Thank you for your time!