The Boustrophedon decomposition method for offline robot complete path planning

Abdou WADE
Supervisors: Luca Berti, Céline Van Landeghem, Christophe Prud'homme

University of Strasbourg

23 août 2023

Table of Contents

1. Introduction
2. Boustrophedon Cellular Decomposition
3. Boustrophedon decomposition path planning with obstacles
4. Conclusion

General context

- **Motion planning** is an essential discipline in robotics and artificial intelligence.

General context

- **Motion planning** is an essential discipline in robotics and artificial intelligence.
- **Path planning** is a subset of Motion planning, which deals with finding an optimal path from point A to point B.

General context

- **Motion planning** is an essential discipline in robotics and artificial intelligence.
- **Path planning** is a subset of Motion planning, which deals with finding an optimal path from point A to point B.
- complete path planning : the robot must visit all the points of the domain.

General context

- The general context of this internship falls within the broad field of motion planning in robotics and artificial intelligence.

Objectives

The goal of this internship is to develop a path planning algorithm for a spherical robot immersed in a fluid.

Objectives

The goal of this internship is to develop a path planning algorithm for a spherical robot immersed in a fluid.
The specific objectives of this project are as follows :

- Implement the **Boustrophedon Decomposition** algorithm.

Objectives

The goal of this internship is to develop a path planning algorithm for a spherical robot immersed in a fluid.
The specific objectives of this project are as follows :

- Implement the **Boustrophedon Decomposition** algorithm.
- Generate a **CSV** file containing the path followed by the robot.

Objectives

The goal of this internship is to develop a path planning algorithm for a spherical robot immersed in a fluid.
The specific objectives of this project are as follows :

- Implement the **Boustrophedon Decomposition** algorithm.
- Generate a **CSV** file containing the path followed by the robot.
- Visualize the robot path and simulate the path using the **Feel++ fluid toolbox**.

Boustrophedon decomposition

The **"Boustrophedon Cell Decomposition (BCD)"**.

Boustrophedon decomposition

The **"Boustrophedon Cell Decomposition (BCD)"**.

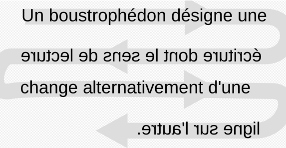


FIGURE – Illustration explaining the principle of a boustrophedon

The robot's trajectory is then determined by the sequence of cells to be visited.

https ://fr.wikipedia.org/wiki/Boustroph

Visualization of the robot's trajectory

To visualize the robot's trajectory, we read the **csv** file of positions

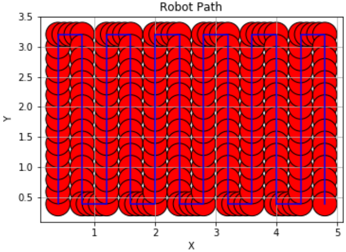


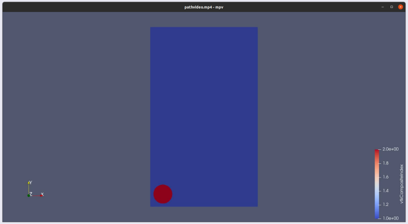
FIGURE – Boustrophedon motion in a rectangular domain without obstacles

Simulate the robot's trajectory

To simulate the robot's trajectory in a fluid, we use the **fluid toolbox** of the **Feel++** finite element library and the **"csv"** file containing the velocities.

Visualization of the robot's trajectory in **Paraview**

2D simulation



Path planning with obstacles

- We have a robot that moves in a 2D environment with obstacles.

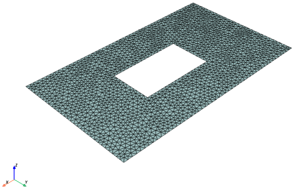


FIGURE – Robot moving in a 2D environment with obstacles

- We want to find a covering the entire domain

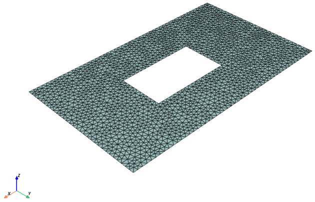
Path planning with obstacles

In this context, the Boustrophedon decomposition algorithm proues to be an effective approach to solving this problem. The algorithm is based on the following steps :

- We start by reading the mesh of the domain.
- Then we do the ray tracing on the mesh.
- We then plot the graph.
- Finally, we visualize the robot's trajectory.

Reading the mesh

Use the **Pyvista** library to read the mesh and display it. In the case of a single obstacle, we have the following figure :



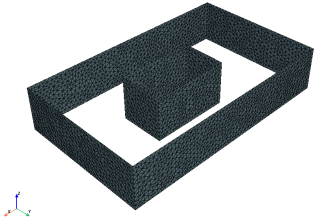


FIGURE – Surface 2D, Extrusion of the boundary of the domains

https ://fr.wikipedia.org/wiki/Extrusion

reading the mesh

And the following figure in the case of multiple obstacles.

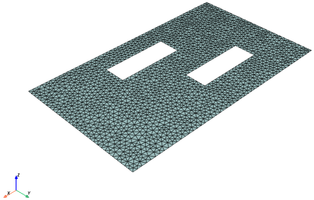


FIGURE – Surface 2D

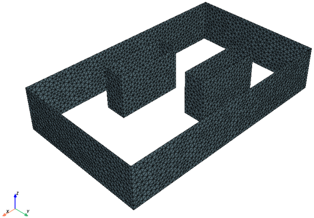


FIGURE – Extrusion of the boundary of the domains

Ray tracing

It's on these surfaces that we did our ray tracing with **ray tracing** to be able to form the graph with the connectivity changes.

Ray tracing

Change in connectivity when the ray touches the obstacle.

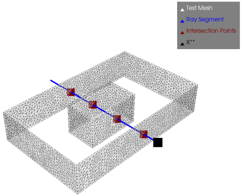


FIGURE – Ray tracing

At $\mathbf{X} = \mathbf{X}^*$, there are 4 intersections, hence 2 cells to closed.

Ray tracing

Change in connectivity when the ray no longer touches the obstacle.

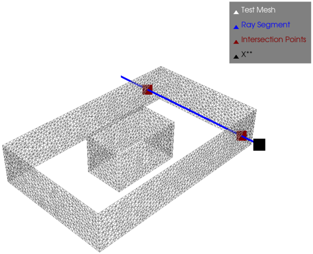


FIGURE – Ray tracing

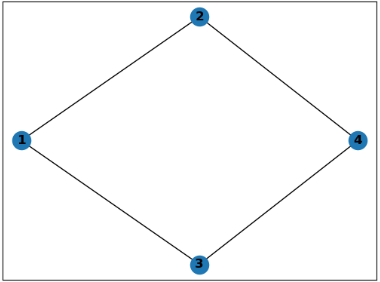
At $\mathbf{X} = \mathbf{X}^{**}$, there are 2 intersections, the previous 2 cells have been closed, a new one opened, and it's still open

Plotting the graph

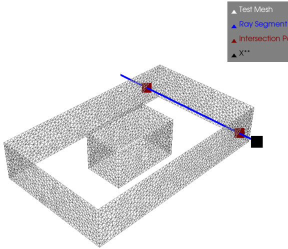
- The domain is decomposed in to 4 cells loch associated to a node of the graph representing cells and their musual connections.
- The nodes are represented by the numbers **(1, 2, 3 and 4)**
- The edges are added to the G graph to connect adjacent cells with the same **xmin** and **xmax** limits.

Graph and ray tracing with a single obstacle

For a single obstacle in the domain, we have the following graph :



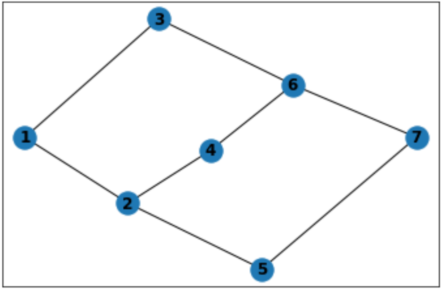
Graph with a single obstacle



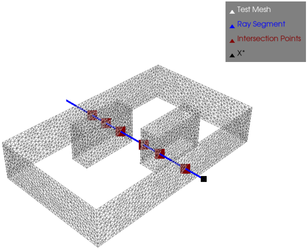
ray tracing

Graph and ray tracing with multiple obstacles

And for multiple obstacles in the domain, we have the following graph :



Graph with multiple obstacles



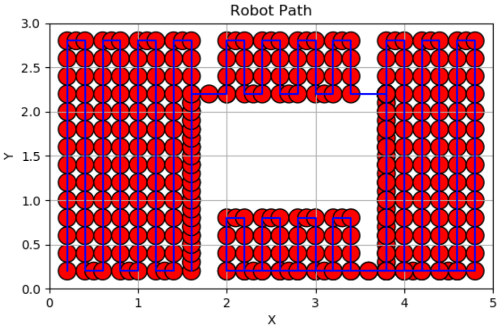
ray tracing

Visualization of the robot's trajectory

- Construction of the **Boustrophedon motion** of the nodes.
- Traversal of the nodes. We then perform a depth-first search (DFS) to traverse the graph.

Visualization of the robot's trajectory

The following figure shows the robot's path through the working area without touching the obstacle.



Conclusion

- We have implemented the Boustrophedon Decomposition algorithm for a mobile robot moving in a 2D environment with obstacles.
- We have also visualized the robot's trajectory and the graph.

Perspectives

And for the perspectives, we have :

- We're also going to change the shape of the obstacles.
- Graph generation optimization.
- Implement the online complete coverage algorithm **BA***.

References I

BA* : an online complete coverage algorithm for cleaning robots,
https ://link.springer.com/article/10.1007/s10489-012-0406-4
https ://github.com/networkx/networkx,
https ://networkx.org/
https ://fr.wikipedia.org/wiki/Boustroph
To launch Rays :
https ://docs.pyvista.org/version/stable/examples/00-load/create-poly.html

Thank you for your attention !