Path Planning in Flow Currents

M.T. Tristão
Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Belo Horizonte, Brasil
marcotuliopin@ufmg.br

***Abstract*—Traditional approaches to solving orientation and path planning problems typically focus on identifying trajectories that optimize an objective function, often related to minimizing travel time. However, the presence of flows in the environment, such as ocean currents or winds, introduces an additional challenge to these approaches. This work proposes a new approach to finding efficient paths in environments with flows. Instead of simply minimizing travel time, the proposed methodology considers the impact of flows on the trajectory, seeking solutions that take advantage of the forces present in the environment to optimize movement.**

I. Introduction

The increasing use of autonomous vehicles (AVs) in marine and aerial environments for scientific and commercial purposes, such as cargo transportation, environmental monitoring, and data collection, has driven the need for efficient path planning algorithms in environments with dynamic flows. This need becomes even more evident when AVs have maximum speeds lower than the flows present in the environment, such as ocean currents or winds. In this context, the development of algorithms that allow AVs to optimize their mission objectives, taking into account the impact of flows on their movements, becomes crucial.

The specific problem addressed in this work is the planning of efficient paths for autonomous vehicles (AVs) in environments with flows. The main challenges of this problem include: Accurately representing the environment's flows, capturing their spatial variations; developing an algorithm that consider the impact of flows on the trajectory. This algorithm is built by creating a discrete representation of the environment with a grid and applying a greedy search to find a feasible path from the origin to the goal.

At the end of this work, we expect to present a robust methodology for planning efficient paths in environments with flows.

II. Related Work

AI The article by Macharet et al. [1] introduces an approach based on the application of a genetic algorithm to find a path that maximizes energy efficiency. Genetic algorithms are a type of optimization and search technique inspired by the process of natural selection, where the fittest individuals are selected for reproduction to produce the offspring of the next generation. In this context, the "fittest" path would be the one that maximizes energy efficiency.

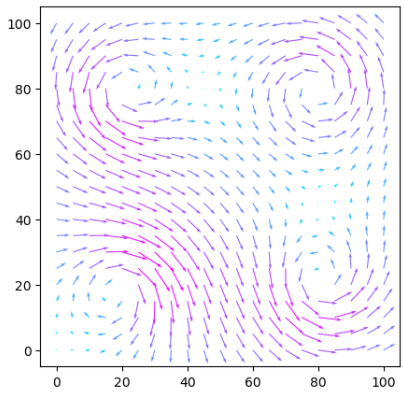

Fig. 1. Environment with flow currents.

On the other hand, Kularatne et al. [2] propose two optimization functions for the graph-based approach: one that minimizes travel time and another that minimizes energy consumption. This dual-objective approach allows for a more flexible path planning strategy that can adapt to different mission requirements or environmental conditions. Importantly, their study also addresses the challenge of time-varying currents by incorporating an additional dimension in the graph representation of the environment to capture this temporal variation. This adds a layer of complexity to the problem but provides a more accurate model of the environment.

Furthermore, Patino et al. [3] and Gunnarson et al. [4] present an approach for the problem in aerial environments, using deep reinforcement learning to find an efficient path. Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with its environment and receiving rewards or penalties. Deep reinforcement learning combines reinforcement learning with deep learning, a type of neural network with several layers, to handle high-dimensional input spaces, such as those encountered in aerial environments.

III. Weight Calculation

The weight calculation for each node in the graph is a crucial part of our path-finding algorithm. The weight, denoted as ω, represents the cost or difficulty of moving to a node, and it is used by the algorithm to determine the most favorable path. In our case, ω incorporates several factors related to

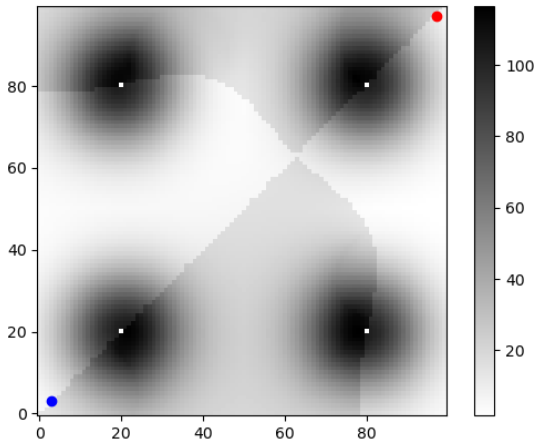

Fig. 2. Weights distribution in the flow map.

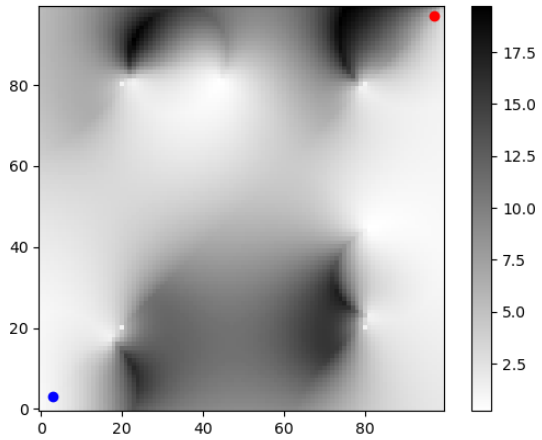

Fig. 3. Mapping of the vortexes orientation in the flow map.

the flow field, the goal direction, and the proximity to vortex centers.

The weight for a node at position (ix, iy) is calculated as follows:

$$\omega = \frac{e^{1+\theta} \cdot ||f_{local}||}{1 + e^{||P||}} + \epsilon \cdot V + \alpha \tag{1}$$

Here's a breakdown of each term in this equation:

- $e^{1+\theta} \cdot ||f_{local}||$: This term represents the influence of the local flow force on the weight. θ is the angle between the goal vector and the local flow force, and $||f_{local}||$ is the magnitude of the flow. This term increases when the flow direction deviates more from the goal direction (larger θ) and/or the flow is stronger (larger $||f_{local}||$). The exponential function $e^{1+\theta}$ ensures that this term grows rapidly for larger angles.
- $1+e^{||P||}$: This term reduces the impact of the above term when the projection of the flow onto the goal vector is large. This encourages the robot to follow the flow when it aligns with the goal direction. In this term, P is the magnitude of the projection of the flow f_{local} onto the goal vector g, where the g is the vector pointing from the current node to the goal.

$$P = \frac{f_{local} \cdot g}{||g||^2} \cdot g \tag{2}$$

- $\epsilon \cdot V$: This term adds a penalty for being close to a vortex center. The V is calculated as in equation 3, where d is the euclidean distance from the current node to the nearest vortex center and β is a constant. This penalty increases rapidly as the robot gets closer to a vortex center, which helps to prevent the robot from getting stuck in a vortex. The term is then weighted with a constant ϵ.

$$V = e^{-d^2/(2\beta^2)} \tag{3}$$

- α: This term adds a penalty if the flow is against the goal direction. If the angle θ is greater than $\pi/2$, indicating that the movement is against the goal, an arbitrary penalty is added to the weight.

This weight calculation balances the need to follow the flow (especially when it aligns with the goal direction), avoid vortex centers, and reach the goal efficiently. By adjusting the weights and factors in this equation, we can fine-tune the behavior of our path-finding algorithm to suit different problem scenarios.

In the figure II the four strong black radil correspond to the vortexes present on the map. These areas must be avoided due to the risk of entrapment by the current. The gray areas represent places where the flow is oriented away from the target point (red dot on the picture).

Additionally, in figure IV, you can observe the effect of the first and second terms of the weight equation. They serve to map the orientation of the vortexes flow. The robot initially avoids going against the flow and prioritizes following the vortex rotation in direction to the goal.

IV. Path Finding Algorithm

This algorithm is a modified version of Dijkstra's shortest path algorithm, which is tailored to handle finding an optimal path in a scene with ambient flows. The goal is to find a path that minimizes the cost while taking advantage of the momentum provided by the ambient flows.

A. Initialization

A priority queue is initialized with the starting node, and two arrays are initialized to store the minimum distance from the start to each cell and the previous cell on the path.

In each iteration of the main loop, a cell is dequeued from. If the dequeued distance is greater than the current minimum distance, the cell is skipped.

B. Calculate Momentum Gain

The alignment between the ambient flow and the direction towards the neighbor is calculated, which is then multiplied by the weight of the neighbor cell. The result of the calculation is then stored in λ. This reflects the principle that movement

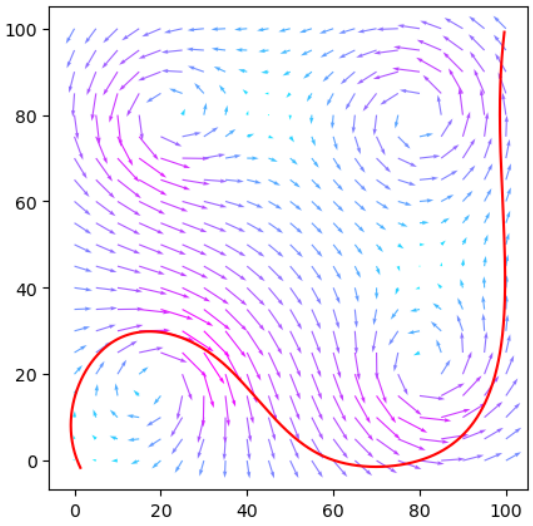

Fig. 4. Path found in the example map.

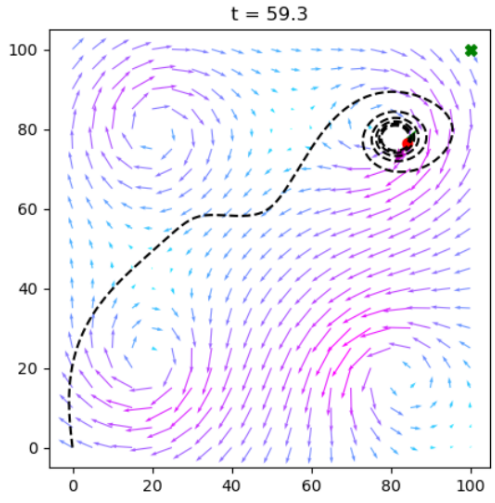

Fig. 6. Robot path following the currents.

in the direction of the flow should incur a lower cost than movement against it. Consequently, this encourages the robot to consider paths that initially diverge from the direction of the goal if such paths result in gaining momentum that ultimately propels it towards the goal.

C. Calculate Edge Force

The angle between the direction towards the neighbor and the local force is calculated using the arc cosine of their dot product. This angle represents the change in direction required to move from the current cell to the neighbor cell, considering the ambient flow at the current cell.

The local force magnitude is simply the magnitude (or length) of the previous force vector at the current cell. It is calculated using the Euclidean norm of the local flow.

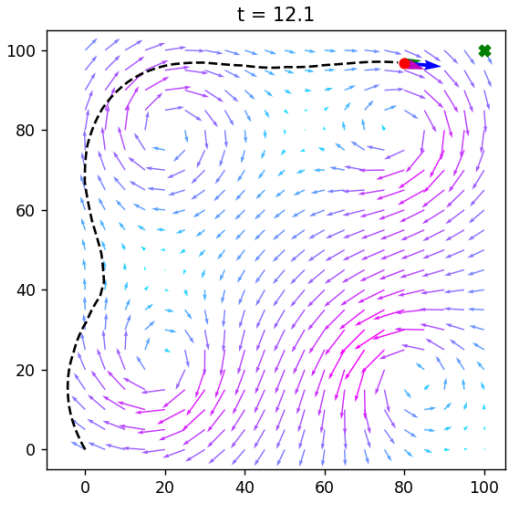

Fig. 5. Robot path using the proposed algorithm.

The edge force f_e is a measure of the cost associated with changing direction when moving from the current cell to the neighbor cell. It is calculated as the exponential of (1 + edge angle) times the local force magnitude. The use of the exponential function ensures that the edge force increases rapidly as the angle increases. This means that sharper turns (larger angles) will have higher costs. Multiplying by the local force magnitude scales the edge force according to the strength of the ambient flow at the current cell.

D. Calculate Resultant Force

The resultant force towards the next neighboring cell is denoted by f_{res}. It is computed as the maximum of two quantities: the magnitude of the attraction force towards the neighbor (provided by the robot's thrust force) minus the local force strength, and zero. This term was incorporated to account for scenarios where a robot with sufficiently strong thrust force can effectively disregard the environmental currents. This value is then stored in r.

E. Update Distance

The new distance to the neighbor cell is calculated as the sum of the current distance, ω and f_e minus λ and r. If this measure is less than the current distance, the distance and the previous cell for the neighbor cell are updated.

The neighbor cell is enqueued into the priority queue for further exploration.

This algorithm finds the shortest path from the start to each cell, considering both the weights and the forces at each cell, as well as an attraction towards the goal. This allows it to find a path that not only minimizes the cost but also takes advantage of the ambient flows and moves towards the goal.

V. Setting the Robot Force

The force exerted by the robot is determined by taking into account both the environmental flow at the robot's current position and an attraction towards the target.

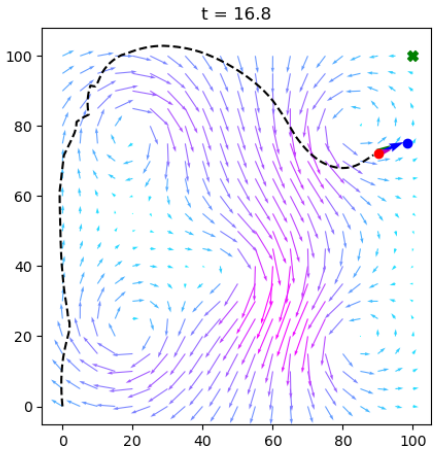

Fig. 7. Robot path using the proposed algorithm.

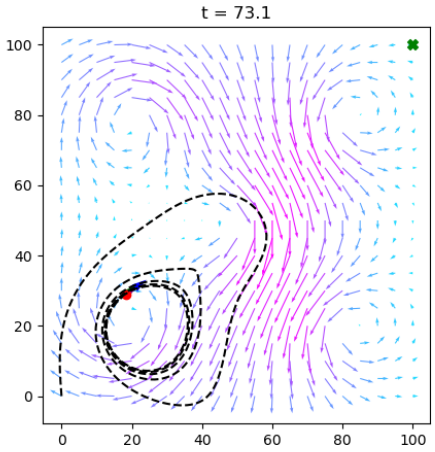

Fig. 8. Robot path following the currents.

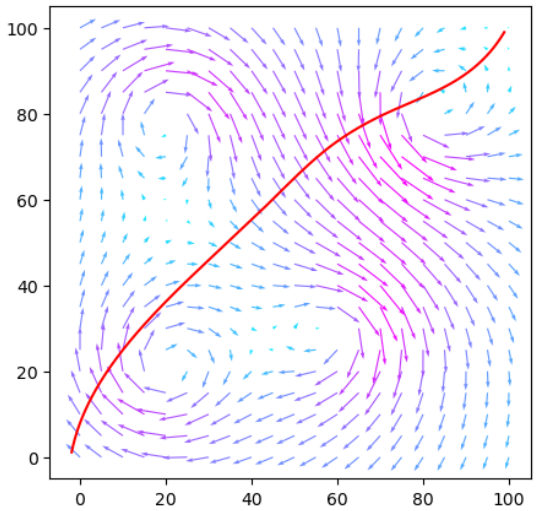

Fig. 9. Path of a robot with strong attraction force.

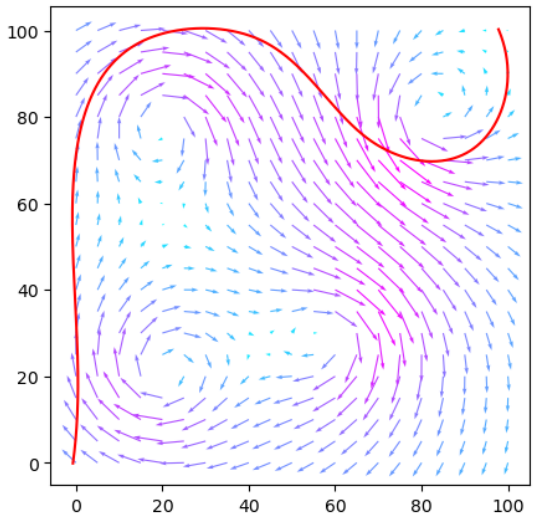

Fig. 10. Path of a robot with weak attraction force.

First, the environmental flow at the robot's current location is calculated. This flow represents the influence of the environment on the robot's movement.

Next, the desired direction towards the target is computed. This is a unit vector pointing from the robot's current position to the target. The target correspond to a point in the path.

An attraction force is then calculated to ensure that the sum of this force and the environmental flow points towards the target. This attraction force represents the additional force that the robot needs to apply in order to move towards the target against the environmental flow.

This attraction force is then normalized to have a constant magnitude. This ensures that the attraction force always has the same strength, regardless of its direction. It is assured that the attraction force of the robot will never have a magnitude greater than the maximum flow of the environment.

Finally, the total force that the robot applies is calculated as the sum of the attraction force and the environmental flow. This total force takes into account both the robot's attraction towards the target and the influence of the environment, providing a comprehensive representation of the forces acting on the robot.

VI. Results

For the map in picture II, our algorithm found the path shown by a red line in figure IV-A. As you can observe, the path uses of the twist motion of the vortexes to go towards the goal, while avoiding going against the current and the vortices centers. A comparative example can be found in figures IV-B and IV-E. The first one shows the path followed by the robot in using the proposed algorithm, while the other shows the path of the robot if it where to simply follow the currents.

In both tests, the thrust force of the robot was the same. We can observe that, by taking advantage of the currents and by

avoiding the vortex centers, the robot was able to get to the objective, while in the other approach the robot got sucked into the center of a vortex. Additionally, figures V and V show another example of how the proposed algorithm finds non trivial paths through environmental flows.

An additional ten scenarios were employed to compare the performance of the two methodologies. The basic approach failed to guide the robot to its target in four out of these ten scenarios, whereas our proposed algorithm successfully navigated the robot to its goal in all instances. In the scenarios where both approaches succeeded, our proposed algorithm reached the goal more rapidly in two cases, while one scenario resulted in a tie with respect to the time taken to reach the goal. These scenarios were generated by altering various parameters of the vortices. This included modifying the positions of existing vortices, introducing new vortices, reversing the rotation of certain vortices, and adjusting the strength of their respective flows.

While the proposed approach may not consistently outperform in terms of speed across all scenarios, it has demonstrated significantly greater reliability under a variety of conditions.

A. Testing Different Robot Thrust Forces

The same scenarios were evaluated using simulations of robots with varying thrust forces. This was achieved by modifying the parameter corresponding to the attraction force. When this parameter is set sufficiently high, the robot possesses enough strength to effectively disregard the environmental forces. This phenomenon is illustrated in figures V and V. The former depicts the path of a robot equipped with a strong attraction force, while the latter represents the path of a robot with a weaker attraction force. These figures clearly demonstrate how the planned path alters when the robot's attraction force is potent enough to overcome the force exerted by the vortex.

VII. Conclusion

In conclusion, this study has presented a novel approach to path planning for autonomous vehicles in environments with dynamic flows. Through the application of our proposed algorithm, we have demonstrated its effectiveness and reliability across a variety of scenarios, showing that it can successfully navigate a robot to its goal even in challenging conditions.

Throughout the course of this work, we encountered several challenges. Accurately representing the environmental flows and capturing their spatial variations was a significant task. Developing an algorithm that could consider the impact of these flows on the trajectory added another layer of complexity to the problem.

Looking forward, there are several ways in which this work could be extended and improved. One potential avenue for future research is to explore other optimization functions for the graph-based approach, such as those that take into account additional factors like energy consumption or safety considerations. Another possible direction is to investigate the use of more advanced machine learning techniques, such as deep reinforcement learning, for path planning in environments with flows.

References

[1] A. Mansfield, D. Macharet, and M. Hsieh, "Energy-efficient orienteering problem in the presence of ocean currents," pp. 10 081–10 087, 10 2022.

[2] D. Kularatne, S. Bhattacharya, and M. A. Hsieh, "Going with the flow: a graph based approach to optimal path planning in general flows," *Autonomous Robots*, vol. 42, pp. 1369–1387, 2018. [Online]. Available: https://doi.org/10.1007/s10514-018-9741-6

[3] D. Patino, S. Mayya, J. Calderon, K. Daniilidis, and D. Saldana, "Learning to navigate in turbulent flows with aerial robot swarms: A cooperative deep reinforcement learning approach," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4219–4226, Jul. 2023, publisher Copyright: © 2016 IEEE.

[4] P. Gunnarson, I. Mandralis, G. Novati *et al.*, "Learning efficient navigation in vortical flow fields," *Nature Communications*, vol. 12, p. 7143, 2021. [Online]. Available: https://doi.org/10.1038/s41467-021-27015-y