ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Semester Thesis

Kinematic coverage planning for object searching with autonomous underwater vehicle

Spring Term 2023

Supervised by:
Florin Kümin
Dr. Rik Bähnemann
Jonas Wüst

Author:
Emilia Szymańska

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

KINEMATIC COVERAGE PLANNING FOR OBJECT SEARCHING WITH AUTONOMOUS UNDERWATER VEHICLE

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
SZYMAŃSKA	EMILIA

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
ZÜRICH, 04.07.2023	Emilia Szymańska

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

Abstract

Coverage path planning, the process of determining a path that ensures the coverage of an entire environment, has found wide application in fields such as surveillance, environmental mapping, and surface cleaning. In situations where certain mobile robots cannot perform on-the-spot turns or where such maneuvers result in longer traversal times, a refined approach is required. This project introduces a solution by applying post-processing, Dubins-like path smoothing to a path generated by a coverage path planner, formulated as a local constrained optimization problem. Subsequently, this smoothed path is transformed into a trajectory. A series of tests on an underwater robot were performed to compare the efficiency of the smoothed trajectory against the original one. The results highlighted the path smoothing algorithm's ability to improve trajectory tracking, reducing the cumulative error by up to 35%, but with some trade-offs in certain situations such as faster in-place rotations. Furthermore, an initial exploration into dynamic path replanning was undertaken, expanding the project's scope. These findings lay the groundwork for future research, aiming to refine these methodologies and pave the way for more efficient smooth coverage path planning.

Chapter 1

Introduction

Robotics is a sector, whose advancements can facilitate or fully automate tasks conventionally performed by humans. The integration of robotic systems can result in an increased repeatability and precision, as well as in cost reduction. Most importantly, robots serve to replace humans in hazardous environments [1], thus improving the safety measures for workers.

One of such robotics-mediated replacement applications lies in the domain of underwater exploration. Elevated pressure, absence of breathable air and the potential threat posed by deep-water animals are just some of the factors that contribute to underwater missions being challenging for human divers. The progress related to remotely operated vehicles (ROVs) or autonomous underwater vehicles (AUVs) [2], equipped with state-of-the-art sensors and peripheral devices, brought significant enhancements in various marine studies. These include comprehensive analysis of marine wildlife [3], the creation of detailed bathymetric maps [4], contribution to marine archaeology [5], or underwater infrastructures maintenance [6].

In the context of robotics, the execution of the aforementioned tasks such as locating ship wrecks, sunken monuments, specific animal species or pipe leakages can be primarily formulated as object detection in a predefined area. To accomplish the task, the environment should be scanned in an efficient manner to avoid unnecessary energy loss and to decrease the search time. The research in coverage path planning for mobile robots addresses this issue [7], aiming to propose a sweeping pattern that not only ensures the full environmental scan, but also optimizes specific characteristics corresponding to the output path (e.g. movement time, number of turns, distance).

The 6 degree-of-freedom design paradigm for underwater robots [8] eliminates the necessity for rotational movements when using bottom-facing scanning sensors. However, the alternative of reducing the number of propellers and simplifying the design encourages engineers to prioritize respectively the decreased costs and improved hydrodynamic properties over the number of degrees of freedom. Hence, the coverage path planning strategies must account for the need to decelerate to a halt, turn in place and re-accelerate to the optimal speed when the objective is the movement time optimization. Another viable strategy would involve implementing path smoothing techniques to take advantages of the robot's kinematic properties [9]. Moving along a curve instead of performing a turn in place could be beneficial in terms of overall search time. However, this approach raises a critical question: how does the post-processing smoothing affect the effectiveness of the coverage?

To answer this question, this study conducted an analytical comparison of outcomes produced when a robot follows either a smoothed or original trajectory. Key performance indicators utilized in this assessment included traversal time, coverage loss, and trajectory deviation.

The smoothing procedure, being a part of the introduced pipeline, was implemented on top of an open-source coverage path planner [10]. The smoothing involved the generation of Dubins path when feasible, with the radius of all circles corresponding to the optimal turning radius of the robot. Depending on the waypoints derived from the coverage path planner, and the obstacles position, the smoothing process detailed in this study considered two primary scenarios. The first scenario, termed *inner smoothing*, transformed three waypoints into a segment-arc-segment path based on a circle inscribed within an angle defined by the waypoints. Conversely, *outer smoothing* was applied in circumstances where *inner smoothing* was unfeasible, resulting in the formation of three arcs. The smoothed path was then converted to a trajectory, which was ultimately forwarded to an underwater robot. An original path consisting only of segments was similarly processed through the trajectory generator.
The considered scenario assumed an object search in an environment with drifting obstacles. To complement the simulated results, tests were conducted with Proteus, a real robot from the Tethys Robotics team. Although the robot had 6 degrees of freedom, its holonomic properties were not fully exploited – scanning was performed with a forward-facing sonar, angled on the robot to ensure adequate bottom coverage. The robot had to follow the whole trajectory; only after the mission completion the object would be identified from the measurements.
The results demonstrated the path smoothing algorithm's effectiveness in enhancing the robot's trajectory tracking capabilities, thus decreasing the cumulative trajectory error by up to 35% for the system under analysis. Interestingly, in-place rotations turned out to be quicker than following a smoothed path due to specific robot characteristics. This unexpected outcome prompted further investigations to better understand and isolate the factors leading to these results. This underscores the importance of considering not just the algorithmic efficiency, but also the intrinsic properties of the robot when evaluating the overall system performance.
In addition, the project proposed an initial version of the dynamic replanner to tackle the challenge of the drifting obstacles. However, the investigated replanning methods were found to be not suited for the improved coverage path planner, primarily due to the exact cellular decomposition integrated in the planner. Further development of post-processing algorithms that can mediate the incompatibilities between the replanner and coverage-handling software is required.

Chapter 2

Related Work

Coverage path planning. The goal of coverage path planning is to find a collision-free path covering the predefined area of interest from the starting point to the final state under optimization conditions such as travel time, number of turns, covered distance etc. The applications of coverage path planning include surface cleaning [11], quality inspection [12], surveillance [13], lawn mowing [14], environment mapping [15] or object searching [16].
Coverage path computation begins with the decomposition of the area of interest into a set of sub-areas. This process can be approached through various methods. For instance, trapezoidal cell decomposition [17], one of the more simplistic techniques, divides the region into trapezoids upon encountering vertices during a line sweeping process. A refined version of this, the boustrophedon cellular decomposition [18], merges the cells which entirely share an edge, thus generating a collection of polygons. Its primary goal with respect to the trapezoidal version is to reduce the number of excessive lengthwise motions and turns. However, it may result in a creation of concave polygons, therefore creating a need for a convex decomposition [19] for some coverage algorithms [20, 21].
There exist numerous strategies to address the coverage path finding problem [7]. Grid-based methods, for example, decompose the targeted region into a set of uniform grid cells. The accomplishment of full coverage is obtained by visiting all these cells with the use of e.g. a wave front algorithm [22]. However, the problem grows linearly in memory in relation to the size of the environment, and exponentially with respect to an optimal path search, which makes the approach unfeasible for large regions of interest and robots with small sensor footprint. The spanning-tree based coverage [23] guarantees the cells to be completely covered, but the generated solutions contains relatively a high number of turns. Additionally, the cell decomposition depends on the sensor's footprint – if it is big enough, numerous cells experiencing partial coverage by obstacles are disregarded, resulting in coverage loss. Randomized algorithms [24, 25], while easy to implement and not requiring robot localization, tend to result in a high area overlap. Artificial potential fields are suitable for real-time planning and proved to be effective in obstacle avoidance [26]. However, a possibility of falling into a local optimum and poor coverage of area close to obstacles indicate a need for further improvement. Numerous other algorithms, including sampling-based planning (rapidly exploring random tree [27], next best view [28]), greedy/graph search (depth-first search [29], breadth-first search [30], Dijkstra's algorithm [31], A* [32]), metaheuristics (genetic algorithm [33], ant colony optimization [34]) and human-inspired techniques (deep reinforcement learning [35], biologically inspired neural networks [36]), have been applied to coverage path planning, trying to enhance specific properties and extending the range of application. Additionally, path interlacing [21], optimal path calculation considering takeoff and

landing points [20], coverage in higher dimensions [37], or addressing the constant environmental disturbances [38] are some of the modifications for consideration when implementing an application-specific solution.
The project presented in this report was build upon a coverage path planning based on boustrophedon cell decomposition and generalized travelling salesman problem [10]. This approach proved to result with even up to 14% lower path costs than conventional coverage planners. As the robot is searching for an object lying on the bottom of the water body, 2D coverage is sufficient, with a controller maintaining the same distance between the sensor and the bottom. Most importantly, in contrary to most research papers, the program associated with the presented method was shared open-source. This allowed for focusing on the path smoothing and dynamic replanning in the case of moving obstacles, which were not implemented in the original tool.
Path smoothing. The incorporation of smooth transitions in path planning is critical for non-holonomic robots, particularly those unable to change their orientation without forward or backward motion. Such smoothing can be performed by employing different curves definition.
A common strategy is the usage of a Dubins (or Reeds-Shepp) path, consisting only straight-line segments and arcs being a part of tangential circles. This relatively simple, yet effective technique has seen extensive usage in coverage path planning solutions [14, 39, 40].
Alternatively, clothoids utilized as primitives for path smoothing [41, 42] offer the benefits of low computational complexity and non-zero initial curvature – features particularly important in replanning. However, the implementation of specific algorithm components can be challenging due to limited available information.
Bezier curves [43], renowned for their few fluctuations and small curvature, simplify the implementation of the control strategy for mobile robots. Although Bezier curves yielded better results than cubic splines and Hermite curves, the track is always contained in the polygon formed by the original path points. This becomes a problematic scenario when an obstacle lies within this polygon, as the algorithm does not perform smoothing on the outer side of this polygon. Obstacle collision issue also arises with splines [44], where the control over the side of the smoothing is limited.
Due these limitations, Dubins path approach was chosen for this project, as it allows for easy extension with additional arcs for obstacle avoidance. Furthermore, while most available programming libraries lack support for collision checking between sophisticated curves and polygons, collision detection between arcs and polygons is widely supported, which further emphasizes the appropriateness of Dubins path application.
Dynamic replanning. Managing drifting or mobile obstacles within an environment is a considerable challenge when avoiding collisions once the coverage path plan is already generated. Additionally, the presence of constant disturbances such as wind or current can result in deviations from the intended path. Such circumstances can lead to coverage loss, especially when obstacles free their initial positions, hence leaving their prior locations unaddressed in the original coverage plan. Therefore, a replanning should be performed, accounting for the already covered area, robot's current position, and updated obstacles locations. While certain coverage path planners introduced real-time replanning [37, 45, 46], the software base selected for this project does not support it. Hence, additional software components were integrated and evaluated in simulation.

Chapter 3

Method

3.1 Method overview

The pipeline implemented in this project is depicted in Fig. 3.1. The utilized coverage path planner requires an input in the form of a polygon with holes, representing an environment with obstacles. The system also requires parameters such as sensor footprint data, cost function type, obstacle offsets etc. Upon receiving these inputs, the planner outputs a set of waypoints corresponding to a computed segment-based path. These waypoints are then processed by a trajectory smoothing generator to create a list of consecutive trajectory poses. If the user specifies a need for it, a replanning procedure is initiated. This process involves forwarding the updated information regarding the environment, which includes the altered positions of obstacles, and a set of already visited poses to the replanning program. It then returns an updated set of trajectory poses. The forthcoming sections of this chapter provide an in-depth discussion on the smoothing procedure, trajectory generation, and the implemented replanning approach.

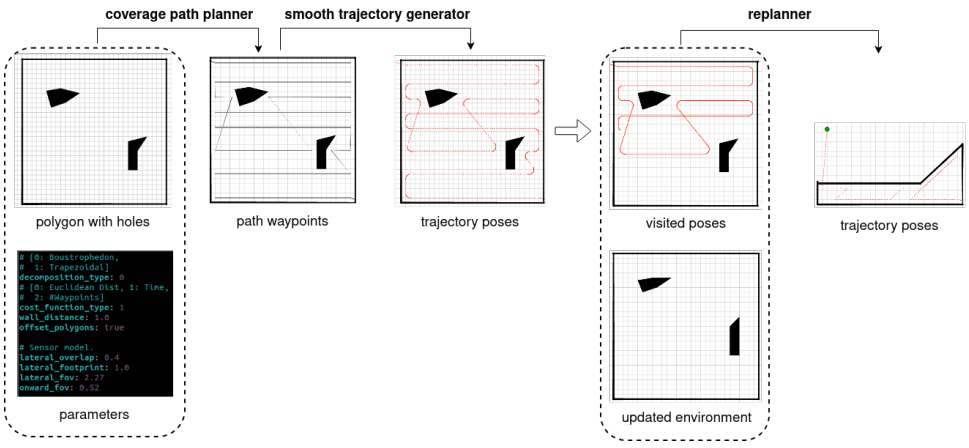


Figure 3.1: Method pipeline.

3.2 Path smoothing

Path smoothing process presented in this project assumes two main scenarios, one of them having two variations (Fig. 3.2).

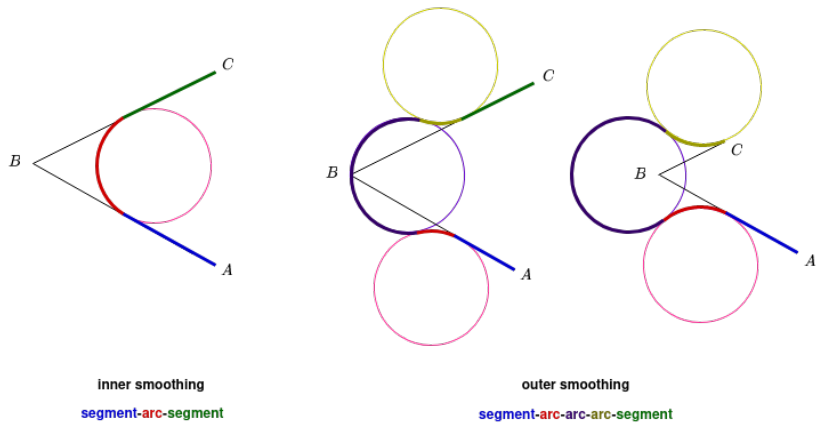
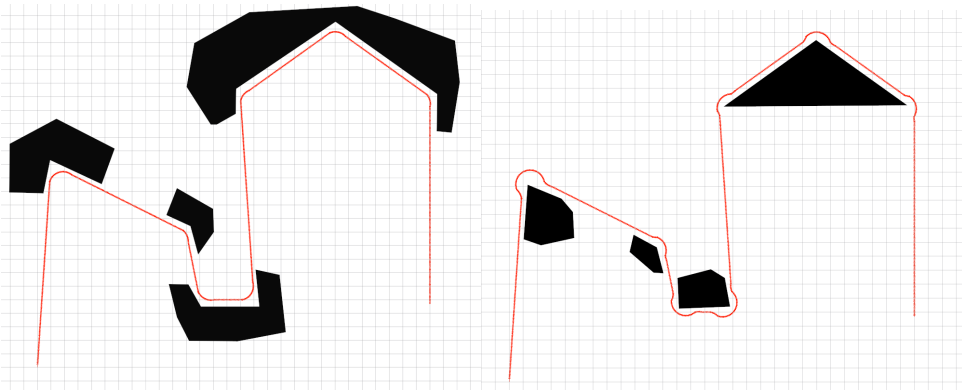


Figure 3.2: Smoothing scenarios.

The first scenario, referred later to as *inner smoothing*, transforms three waypoints into a configuration sement-arc-segment. The arc is derived from a circle inscribed within an angle, shaped by the said trio of waypoints, with a start and end point being intersection points with angle's arms.
On the other hand, *outer smoothing*, applied in situations when the inner one is not feasible (e.g. obstacle lies on the inner side of the angle), is formed by three arcs. When it is possible, the middle arc is placed in the proximity of to the middle waypoint. However, if such an arrangement results in a collision with any obstacle or in the other waypoints being too close to the outer arcs (for them to lie within the segments), the circles are repositioned accordingly. These two cases are referred to as *minimum at bisector boundary* and *minimum at tangent intersection boundary* respectively.
The example of applications of different smoothing variants is presented in Fig. 3.3.



(a) Inner smoothing (b) Outer smoothing

Figure 3.3: Example of applications of the two versions of smoothing.

The smoothing procedure, as outlined in Alg. 1, consecutively evaluates the feasibility of each of the smoothing scenarios, with inner smoothing being prioritised over the outer one. The program's main loop iterates over the waypoint list returned by the coverage path planner. The input values of the algorithm are three waypoints, forming a convex angle, and an optimal turning radius. The algorithm returns a pair of a binary value signifying the procedure's success status, and a set of output

arcs. If the smoothing is not possible to be performed with the input parameters, the corresponding phase of the path is filled with straight line segments. Additional segments, which join the arcs from distinct smoothing phases, are added through a separate procedure.

Algorithm 1 Compute smoothing

Input: $A, B, C \in$ geometry_msgs::Pose, $r > 0$, $\angle ABC \in (0; \pi)$
  Convert A, B, C to CGAL::Point_2
  $arcs \leftarrow inner\_smoothing(A, B, C, r)$
  if not $feasible(arcs)$ **then**
    $arcs \leftarrow outer\_smoothing\_min\_bisector(A, B, C, r)$
    if not $feasible(arcs)$ **then**
      $arcs \leftarrow outer\_smoothing\_min\_tan\_intersection(A, B, C, r)$
      if not $feasible(arcs)$ **then**
        return $(success = false, arcs)$
      end if
    end if
  end if
  return $(success = true, arcs)$

What is important to emphasise, the first waypoint considered in each smoothing does not necessarily corresponds to the one generated by the coverage path planner. It may be shifted to the position of the end point of the final arc derived from the preceding smoothing iteration. Consequently, new smoothing sequence's arc can begin no sooner than the previous smoothing ends. This technique ensures path continuity and maintains a logical progression through the multiple smoothing phases.
The feasibility assessment of each smoothing scenario comprises two distinct phases. The first phase ensures path continuity by analysing the proximity between consecutive segments and sets of arcs. The second phase involves iterating through all obstacles' edges, verifying the lack of collision between them and the generated arcs.

3.2.1 Inner smoothing

The smoothing process as proposed in this study can be conceptualized as a local constrained optimization problem. The ultimate objective of this optimization process is to minimize the path length and to maintain a minimal divergence between the smoothed path and the original one, with the path being smoothed using Dubins curves.
In scenarios involving inner smoothing, this objective can be accomplished through the consideration of a single arc. The optimal solution for the arc's origin is subject to a constraint, that the path needs to be continuous, in terms of both the strictly geometrical aspect and the robot's orientation changes (the derivative of the robot's orientation with respect to the progression along the path, must be finite and should not exhibit abrupt changes).
In accordance to the symbol assignments presented in Fig. 3.4, the constraint can be formally encapsulated by a set of equations as shown in Eq. 3.1.

$$\begin{cases} r = dist(\overline{\mathrm{AB}}, O) \\ r = dist(\overline{\mathrm{BC}}, O) \end{cases} \tag{3.1}$$

Taking these constrains into consideration, the inner smoothing method relies on

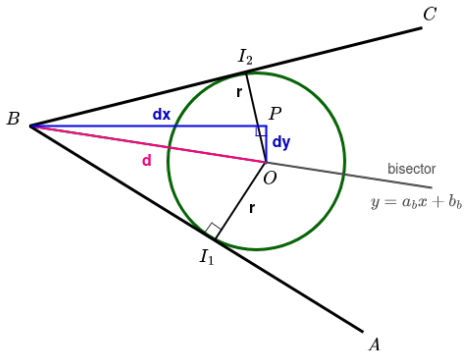


Figure 3.4: Trajectory from path

the determination of a circle inscribed in an angle formed by the waypoints. The output arc is formed by the intersection points I_1 and I_2.
The determination of the circle's center requires the coordinates of the three waypoints A, B, C and the circle's desired radius. The trigonometric relationships in $\triangle BI_1O$ and $\triangle BPO$ allow to compute dx, dy values. These values, when combined with B point coordinates, define the position of O, which is required to lie on the bisector of $\angle ABC$. By utilizing the law of cosines depicted in Eq. 3.2 to obtain $\angle ABC$, $\angle ABO$ can be determined, as it is equal to bisecting $\angle ABC$.

$$|AC|^2 = |AB|^2 + |BC|^2 - 2 \cdot |AB| \cdot |BC| \cdot cos\angle ABC \tag{3.2}$$

With the established knowledge of $\angle ABO$ and r, the value of d can be derived from sinusoidal dependency. The angle of bisector's slope corresponds to $\angle OBP$, therefore facilitates the correlation of dx, dy with the value of d. As the bisector is common to both the convex and concave versions of $\angle ABC$, two potential solutions should be considered and the closer one to $\overline{\text{AC}}$ should be selected.
The program-like description of these steps is presented in Alg. 2.

Algorithm 2 Compute the center of the circle inscribed in an angle

Input: $A, B, C \in CGAL :: Point\_2$, $r > 0$, $\angle ABC \in (0; \pi)$
$\alpha \leftarrow \angle ABO = \frac{1}{2} \cdot \angle ABC = \frac{1}{2} \cdot angle(A, B, C)$
$\beta \leftarrow \angle OBP = bisector\_slope\_angle(A, B, C)$
$d \leftarrow \frac{r}{sin\alpha}$
$dy \leftarrow d \cdot sin\beta$
$dx \leftarrow d \cdot cos\beta$
$O_1 \leftarrow Point\_2(x_B + dx, y_B + dy)$
$O_2 \leftarrow Point\_2(x_B - dx, y_B - dy)$
$d_{O_1} \leftarrow distance(O_1, \overline{\text{AC}})$
$d_{O_2} \leftarrow distance(O_2, \overline{\text{AC}})$
if $d_{O_1} > d_{O_2}$ **then**
    return O_2
else
    return O_1
end if

Upon establishing the circle's position, the intersection points with both arms of $\angle ABC$ are computed. The crucial aspect of inner smoothing procedure is to determine the direction of the arc, which is calculated by leveraging the cross product of

vectors $\overrightarrow{AB}$ and $\overrightarrow{BC}$. If its value is positive, the arc is interpreted as counterclockwise; conversely, a negative value signifies a clockwise direction. In both cases, the direction of the arc is opposite to the direction of the directed angle $\angle ABC$.
Finally, an arc characterized by the computed parameters is returned, as shown in Alg. 3.

Algorithm 3 Compute the inner smoothing

Input: $A, B, C \in$ CGAL::Point_2, $R > 0$, $\angle ABC \in (0; \pi)$
$O \leftarrow circle\_inscribed\_in\_angle(A, B, C, r)$
$I_1 \leftarrow intersection(circle(O, r), \overline{\text{AB}})$
$I_2 \leftarrow intersection(circle(O, r), \overline{\text{BC}})$
$dir \leftarrow arc\_direction(A, B, C)$
return arc(start=I_1, end=I_2, center=O, radius=r, direction=dir)

3.2.2 Outer smoothing

The outer smoothing process involves the determination of three circles; the middle one lies on the bisector of angle $\angle ABC$, while the others are tangential to both the middle circle and to angle's arms. Notably, if the points A or C are too close to B for a central circle to consist it, the circles are shifted accordingly. Both procedures, with the formulated constrained optimization, are described in detail in their respective sections.
Similarly to the inner smoothing scenario, the procedure requires the coordinates of the three waypoints A, B, C and the desired radius. It should be noted that the radii of all the arcs are assumed to be the uniform; variable radii would generate an overwhelming number of scenarios, thus exceeding the scope of this project.

Minimum at bisector boundary

In the optimization process of this scenario, it is necessary to balance the objectives of maintaining minimal path length while achieving minimal deviation from the original unsmoothed path. This compromise ensures preserving the overall coverage to a substantial extent. Consequently, the established inequality constraint is that the middle circle's center can only lie along the bisector up to a distance equal to the predefined radius from the central waypoint. In this case, the intersections of the tangential circles can manifest at any point on the segments being the angle's arms formed by the three waypoints. These constraints are mathematically represented in Eq.3.3. The symbolic notation and an illustrative representation of the optimal solution for this scenario is provided in Fig.3.5, with the middle circle's origin lying exactly at the distance of r from point B.

$$\begin{cases} r \geq |BO_2| \\ |O_1O_2| = 2r \\ |O_2O_3| = 2r \\ r = dist(\overline{\text{AB}}, O_1) \\ r = dist(\overline{\text{BC}}, O_3) \end{cases} \tag{3.3}$$

It can be observed that the determination of middle circle's center should be analogous to the inner smoothing procedure, as described in Alg. 4. However, this time d is known and equals r, thereby eliminating the need for the computation steps associated with d.

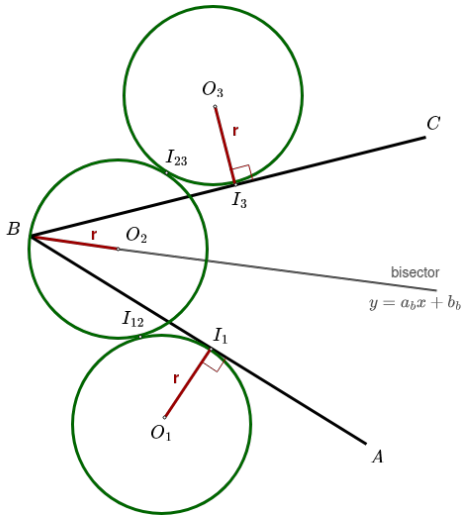


Figure 3.5: Outer smoothing - *minimum at bisector boundary* version

Subsequently, the location of the other circles needs to be found. Given their symmetrical positioning with respect to each other along the bisector, the computation of one circle suffices – the other can simply be mirrored along the bisector's line. For this project, it is assumed that O_3, the last of the three circles, is computed initially.
To calculate it, the positions of all three waypoints, the location of the middle circle and the turning radius are required. The resulting circle has to meet two requirements – it should be tangential to middle circle (with a distance $|O_2O_3|$ equivalent to twice the radius length) and tangential to the angle's arm (with a distance between O_3 and the segment $\overline{\text{BC}}$ equal to radius length). These conditions, after obtaining the parameters of the line $\overline{\text{BC}}$, form a system of equations as illustrated in Eq. 3.4. The system yields four distinct results as presented in Fig. 3.6. To obtain the desired x, y coordinates, first the circles lying on the same side of $\overline{\text{BC}}$ as O_2 are removed. It is accomplished by comparing the directions of vectors created by each circle's center and the closest point on $\overline{\text{BC}}$ – they should be opposite to the vector corresponding to O_2. From the remaining solutions, the one closest to $\overline{\text{AC}}$ is selected. The first circle is then determined to be the mirror image of the obtained O_3 along the bisector.

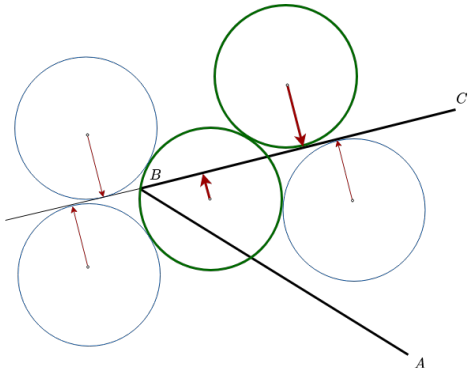


Figure 3.6: All solutions for the system of equations in Eq. 3.4.

The smoothing procedure requires the determination of the start and end points

Algorithm 4 Compute the last circle in minimum bisector boundary for outer smoothing

Input: $A, B, C, O_2 \in CGAL :: Point\_2$, $r > 0$, $\angle ABC \in (0; \pi)$
$a, b, c \leftarrow line\_parameters(B, C)$
$solutions \leftarrow$ solve for x, y

$$\begin{cases} 2r = \sqrt{(x - x_{O_2})^2 + (y - y_{O_2})^2} \\ r = \frac{|ax_{O_2} + by_{O_2} + c|}{\sqrt{a^2 + b^2}} \end{cases} \tag{3.4}$$

for sol in $solutions$ **do**
  if not $on\_opposite\_side(sol, O_2, \overline{\mathrm{BC}})$ **then**
    $solutions \leftarrow remove(sol, solutions)$
  end if
end for
return $closest\_to\_segment(solutions, \overline{\mathrm{AC}})$

of each arc. Consequently, points I_1 and I_3 are computed as the intersection of circles with the arms of the angle $\angle ABC$, while points I_{12} and I_{23} represent points of intersection between circles. The direction of the middle arc is obtained with the same method as in the inner smoothing scenario. The other circles should be oriented in the opposite direction to the middle circle. Ultimately, all three arcs are created using the provided parameters and the triplet is returned by the procedure, as demonstrated in Alg. 5.

Algorithm 5 Compute the outer smoothing in the minimum bisector boundary

Input: $A, B, C \in$ CGAL::Point_2, $R > 0$, $\angle ABC \in (0; \pi)$
$O_2 \leftarrow outer\_smoothing\_middle\_circle(A, B, C, r)$
$O_3 \leftarrow outer\_smoothing\_last\_circle(O_2, A, B, C, r)$
$O_1 \leftarrow mirrored\_point(O_3, bisector(A, B, C))$
$I_1 \leftarrow intersection(circle(O_1, r), \overline{\mathrm{AB}})$
$I_3 \leftarrow intersection(circle(O_3, r), \overline{\mathrm{BC}})$
$I_{12} \leftarrow intersection(circle(O_1, r), circle(O_2, r))$
$I_{23} \leftarrow intersection(circle(O_2, r), circle(O_3, r))$
$dir_2 \leftarrow arc\_direction(A, B, C)$
$dir_1 \leftarrow opposite(dir_2)$
$dir_3 \leftarrow opposite(dir_2)$
$a_1 \leftarrow$ arc(start=I_1, end=I_{12}, center=O_1, radius=r, direction=dir_1)
$a_2 \leftarrow$ arc(start=I_{12}, end=I_{23}, center=O_2, radius=r, direction=dir_2)
$a_3 \leftarrow$ arc(start=I_{23}, end=I_3, center=O_3, radius=r, direction=dir_3)
return (a_1, a_2, a_3)

Minimum at tangent intersection boundary

This particular case is executed when at least one of the points A/C is situated too close to B. The middle circle's origin O_2 lies anywhere on the bisector, though not necessarily in the proximity of B. However, the constraint where the tangent intersection between any other O_1/O_3 circle and angle's arm lies anywhere on the segment $\overline{\mathrm{AB}}/\overline{\mathrm{BC}}$ respectively cannot be met. Hence, the conceptual foundation for this smoothing involves moving the circle's intersection point with the angle's arm, corresponding to the shorter segment, to point A/C. The scenario with $|BC|$ being shorter than $|AB|$ is presented in Fig. 3.7 – I_3 is shifted to C. The updated

constrains are depicted in Eq. 3.5.

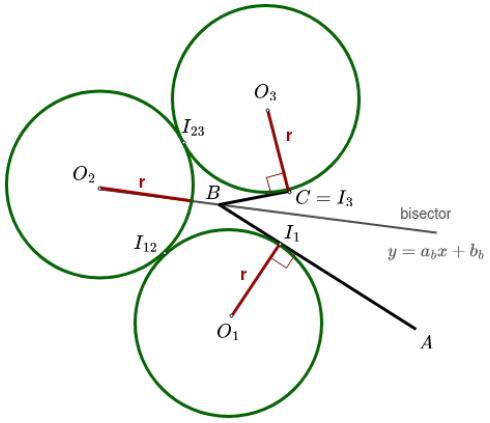


Figure 3.7: Outer smoothing – minimum at tangent intersection boundary version.

$$\begin{cases} |O_1O_2| = 2r \\ |O_2O_3| = 2r \\ |O_1O_3| > 2r \\ r = dist(\overline{\mathrm{AB}}, O_1) \\ r = dist(\overline{\mathrm{BC}}, O_3) \end{cases} \tag{3.5}$$

Firstly, the segments lengths are compared to determine which circle presents more difficulty in fitting into the path, therefore which one should be computed first. This circle has to meet two requirements – it must be tangential to the angle's arm (the distance between the line containing the segment and the center of the circle equals radius length) and its center has to lie on the line perpendicular to the arm of the angle passing through the corresponding waypoint (the equation for this line, when substituted with the circle's center coordinates, must hold true). Both these conditions form a system of equations presented in Eq. 3.6. Once the perpendicular line's parameters are determined, the system can be solved for x, y. There are two potential solutions, symmetrical to each other with respect to the considered segment. The solution furthest from the waypoint on the opposing arm should be selected, as presented in Alg. 6. The center of other circle is obtained by mirroring this solution along the bisector.

Algorithm 6 Compute the begin circle in minimum at tangent intersection boundary outer smoothing

Input: $p_1, p_2, p_3 \in CGAL :: Point\_2$, $r > 0$, $\angle ABC \in (0; \pi)$
$a, b, c \leftarrow line\_parameters(p_1, p_2)$
$a_p, b_p, c_p \leftarrow perpendicular(a, b, c, p_1)$
$solutions \leftarrow$ solve for x, y

$$\begin{cases} 0 = a_p x + b_p y + c_p \\ r = \frac{|ax+by+c|}{\sqrt{a^2+b^2}} \end{cases} \tag{3.6}$$

return $furthest\_from\_point(solutions, p_3)$

It is worth noting that the middle circle is expected to lie on the bisector (the equation for the bisector's line, when substituted with the circle's center x, y, must

hold true) and it should be tangential to any of the other circles (the distance between its center and the center of the other circle must be equalivalent to twice the radius length). Solving this system of equations, illustrated in Eq. 3.7, yields two solutions symmetric along $\overline{O_1O_3}$, and the solution furthest from $\overline{AC}$ should be selected. This procedure is depicted in Alg. 7.

Algorithm 7 Compute the middle circle in minimum at tangent intersection boundary outer smoothing

Input: $O_1, A, B, C \in CGAL :: Point\_2$, $r > 0$, $\angle ABC \in (0; \pi)$
$a, b, c \leftarrow bisector\_line\_parameters(A, B, C)$
$solutions \leftarrow$ solve for x, y

$$\begin{cases} 0 = ax + by + c \\ 2r = \sqrt{(x - x_{O_1})^2 + (y - y_{O_1})^2} \end{cases} \tag{3.7}$$

return $furthest\_from\_segment(solutions, \overline{AC})$

It is important to be ensure that the first and last circle do not intersect each other (which would lead to an excessive coverage overlap) and that they are not positioned too distantly for the middle circle to not fit. This is achieved by verifying the condition $4r > |O_1O_3| > 2r$ prior to generation of the smoothing solution.
The general smoothing steps are demonstrated in Alg. 8.

Algorithm 8 Compute the outer smoothing in minimum at tangent intersection boundary scenario

Input: $A, B, C \in CGAL :: Point\_2$, $r > 0$, $\angle ABC \in (0; \pi)$
if $|AB| < |BC|$ **then**
$O_1 \leftarrow begin\_circle\_outer\_smoothing(A, B, C)$
$O_3 \leftarrow mirrored\_point(O_1, bisector(A, B, C))$
else
$O_3 \leftarrow begin\_circle\_outer\_smoothing(C, B, A)$
$O_1 \leftarrow mirrored\_point(O_3, bisector(A, B, C))$
end if
if $4r > |O_1O_3| > 2r$ **then**
$O_2 \leftarrow middle\_circle\_outer\_smoothing(O_1, A, B, C)$
further arc generation similarly to Alg. 5
end if

3.3 Trajectory generation

Once the path composed of segments and arcs is defined, a corresponding trajectory is generated. In this project, it is assumed that this task is accomplished through the path sampling with a specified interval between each trajectory point. The corresponding timestamps can be calculated from the robot's velocity model when needed.
Program users are required to input two parameters – the sampling distance for straight segments and for arcs, as executing a turn may necessitate higher precision (and hence, more waypoints) compared to traversing a straight path. Subsequently, the poses containing x, y coordinates and orientation quaternion are computed.
An illustrative example of trajectory generation derived from a path is depicted in Fig. 3.8.

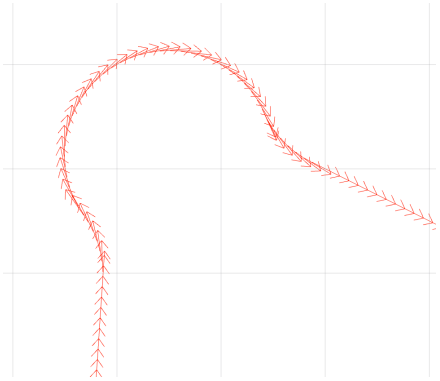


Figure 3.8: An example of trajectory generated from a path, with red arrows representing robot's position and orientation.

3.4 Dynamic replanner

As this project considers the scenario of an underwater robot, the dynamic, aquatic environment requires flexibility in planning. The existing currents can result in obstacles' displacement and cause the robot to drift from the intended path. These cases necessitate the trajectory replanning to accommodate the new conditions.

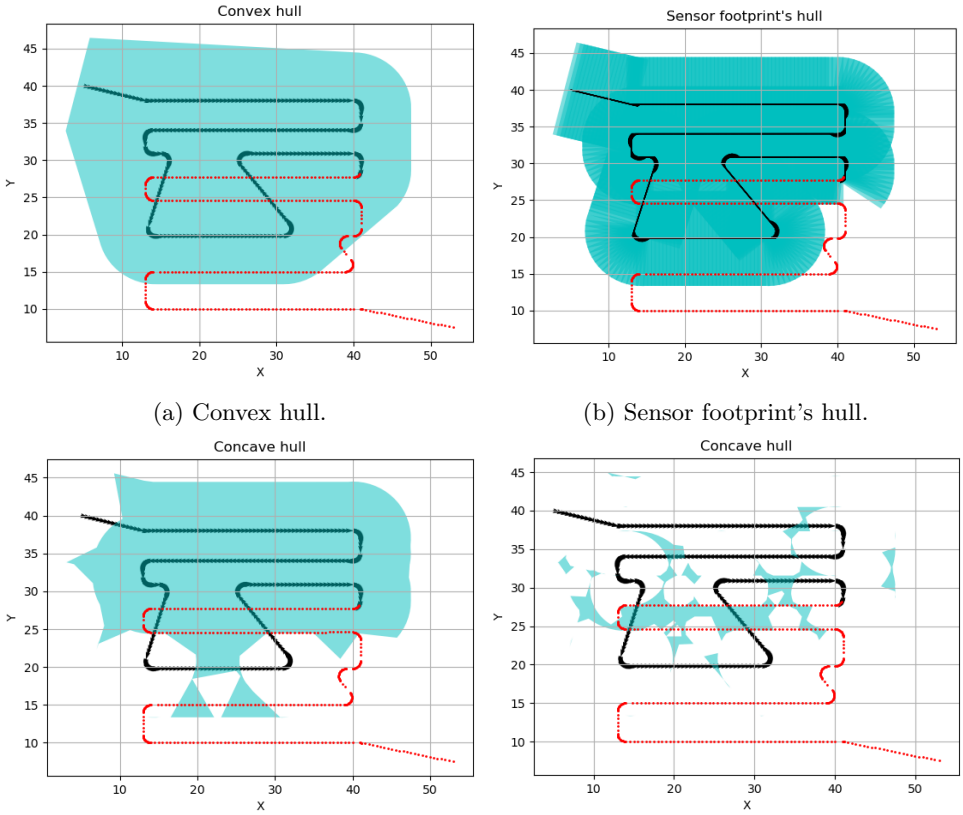
Upon processing information on the updated map of the environment and the historical pose data, the program identifies the remaining area to be covered and the new starting point, which correspond to the last visited pose. The updated data is fed into the coverage planner path recomputation.

Three methodologies have been investigated to determine the remaining area to be covered. Each method requires converting each visited pose into a rectangular representation equivalent to the sensor's footprint at the given robot position and orientation.

The first approach (Fig. 3.9a) relies on the convex hull generation from the vertices of all the calculated footprint rectangles. Upon obtaining the updated polygon with specified holes, the polygon corresponding to the convex hull is subtracted from the environmental map. The planner perceives it as either an obstacle or an area out of interest. However, this approach proves to be restrictive as it might erroneously discard unvisited regions, depending on the position of the vertices.

To address this limitation, the implementation of concave hulls (Fig. 3.9c,3.9d) was investigated. Alpha shapes were used to formulate the concave hull. A crucial aspect of this approach lies in fine-tuning the α parameter, which controls the detail level of the shapes. However, the ideal alpha value is scenario-dependent and an automatic, real-time parameter tuning was found to be unfeasible in this project scope.

The approach that yields the most promising and accurate results involves subtracting each footprint-related rectangle from the input polygon individually (Fig. 3.9b). While this technique does not require parameter tuning or does not discard uncovered areas, it also presents a set of challenges. The approach tends to account for even minimal uncovered areas, resulting e.g. from smoothing-related coverage loss, thereby generating unnecessary path segments or, occasionally, leading to errors due to the polygon not being simple. Moreover, the resulting polygon may contain an excessive number of vertices, especially when a turn is executed and contributes to the polygon's boundary definition. Consequently, the coverage planner may decompose the polygon into numerous parts with very small areas, prompting the robot to sweep them, although due to their proximity is not necessary.



(a) Convex hull.

(b) Sensor footprint's hull.

(c) Concave hull with $\alpha = 0.3$.

(d) Concave hull with $\alpha = 1.0$.

Figure 3.9: Replanning hulls, visualized by blue areas. Black arrows indicate the already visited poses, while the red dots correspond to x, y coordinates of path points that robot would have yet to visit.

The example of unsmoothed replanning performed on the updated environment (Fig. 3.10) is presented in Fig. 3.11. It can be noticed that the convex hull discarded more area than the sensor footprint-based hull. The new plan could not be created for the concave hull case – the generated polygon is not simple, as illustrated in Fig. 3.12.

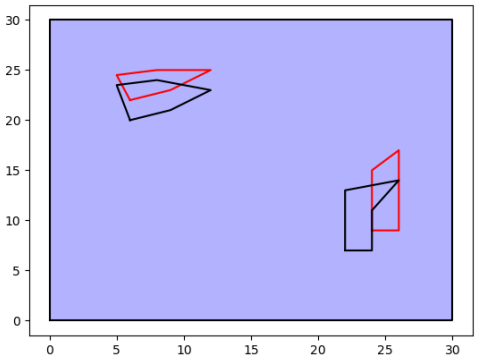
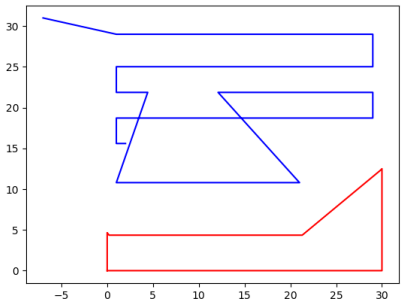
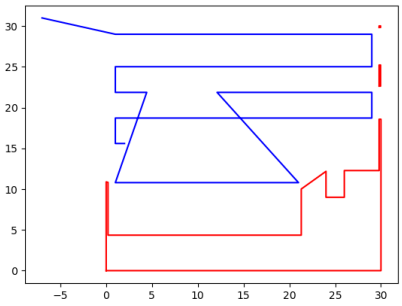


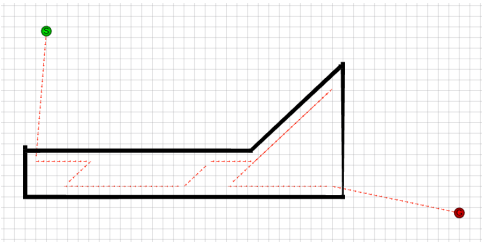
Figure 3.10: The map of the environment with the initial obstacles position (black-bordered polygons) and the updated ones (red-bordered polygons).



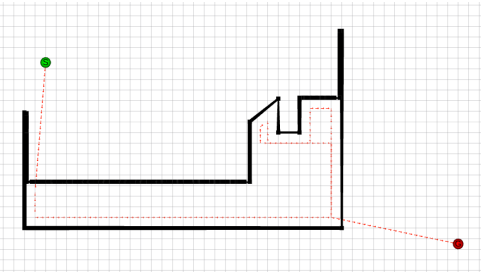
(a) Convex hull replanning.



(b) Sensor footprint's hull replanning.



(c) Generated trajectory for convex hull replanning.



(d) Generated trajectory for sensor footprint's hull replanning.

Figure 3.11: Replanning example for the convex and sensor footprint hull. In a) and b) the blue line represents the traversed path, while the red polygons are the area yet to be covered by the robot.

Potential improvements of the replanning method are discussed in Conclusions. However, due to the limited scope of the project, further development of the replanner was not continued.

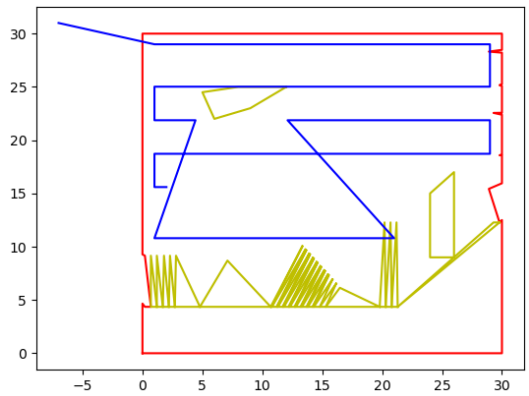


Figure 3.12: Visualization of the problem present with concave hull subtraction. The blue line represents the traversed path, the red polygon is the area yet to be covered by the robot and yellow elements represent the polygon's holes.

3.5 Software pipeline overview

The implemented solution, written in C++, utilizes the ROS Noetic framework, leveraging the open-source coverage path planner [10]. Fig. 3.13 depicts the developed ROS pipeline. Nodes *replanner* and *smooth_trajectory_generator* were integrated into *trajectory_generation_tools* package.
The smoothing logic is implemented in the *smooth_trajectory_generator* node. The node subscribes to the *polygon* topic to contain the obstacles' description for collision-related path assessment, and to *waypoint_list* to access coverage path planner output. The result in a form of trajectory points is published on *trajectory* topic. The node *replanner* acquires the updated map of the environment and the historical pose data from the *updated_polygon* and *visited* topics respectively.
While *coverage_planner* was initially a part of *polygon_coverage_ros*, the start and goal points selection was tedious – the Point messages were published on the same topic and the order defined which message corresponded to the start, and which to the goal. Additionally, once the end point was received, the coverage path planner was immediately activated, without offering an option to change anything in case of an erroneous click in rviz tool or a misspelling in *rostopic pub* command. To provide the user with a more flexible interface, an improved *rviz_point_tool* was implemented (Fig. 3.14). Therefore, modifications were applied to *coverage_planner* interface and for the simplicity of implementation, this node was incorporated into *trajectory_generation_tools* package.
The utilized coverage path planner was developed using C++ language within ROS Noetic architecture. The program further incorporated the Computational Geometry Algorithms Library, and a memetic solver for the generalized traveling salesman problem. To invoke the planner's operation, an input polygon with holes (namely the area of interest) as well as the start and goal positions need to be published on the corresponding topics – or selected with the provided RVIZ user interface. The planner generates a list of waypoints defining an ouput path.
Additionally, the system includes a set of adjustable parameters tailored to user's specifications. For instance, the type of decomposition, cost function, wall offset, or sensor-related settings are some of the customizable inputs. The selected configuration for this project is further described in Chapter 4.

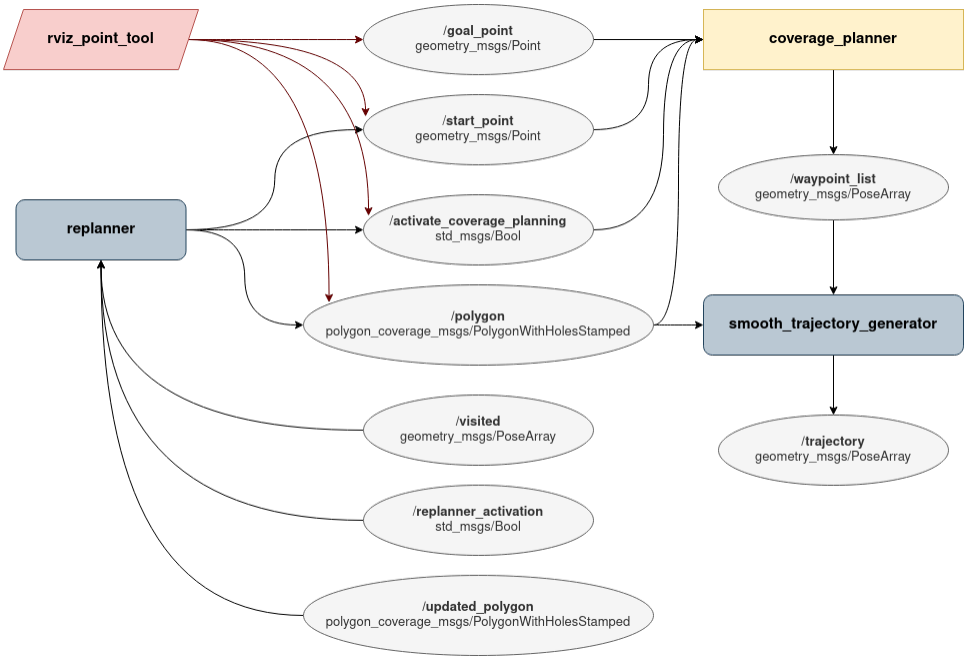


Figure 3.13: ROS pipeline.

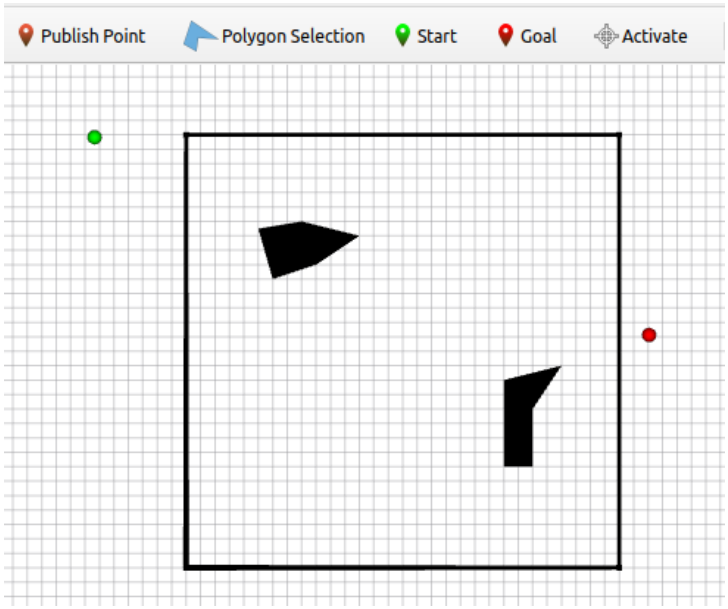


Figure 3.14: Improved RVIZ tool part – addition of *Start*, *Goal* and *Activate* functionalities. The start and goal positions are visualized after being defined with the corresponding tool and can be modified before activating the coverage path planning, in contrary to the scenario with the original *Publish Point* usage.

Chapter 4

Experiments

To evaluate the effectiveness of the proposed smoothing algorithms, a series of tests were conducted using a real-wolrd robotic system – Proteus underwater vehicle, developed by Tethys Robotics team – in the Zurich Lake. The coverage path planner was deployed on a map 30m×30m that featured two obstacles, as illustrated in Fig. 4.1.

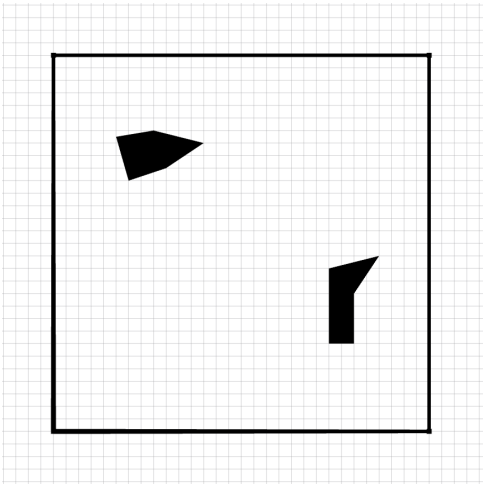
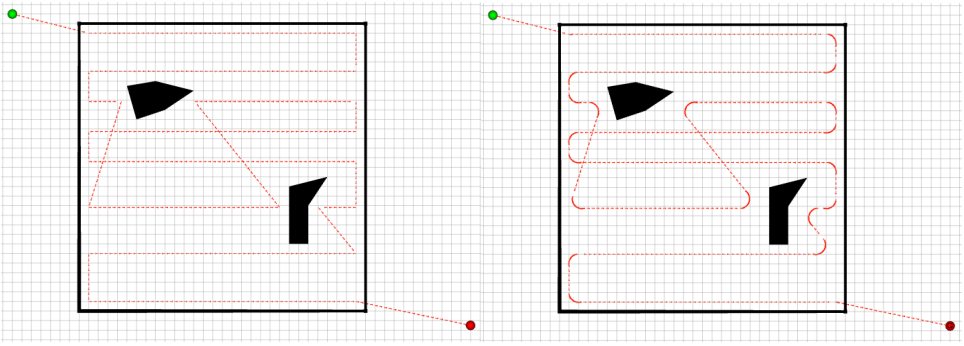
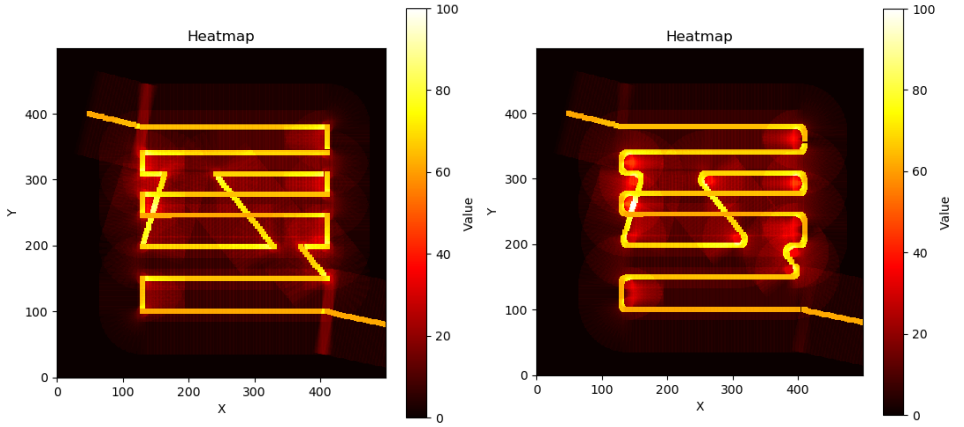


Figure 4.1: The map of the environment for the experiments.

The planner's parameters were set as follows: Boustrophedon decomposition type, time cost function, and a wall offset of one meter. The desired lateral overlap was established at 40%, and the robot was supposed to maintain the distance of 3 meters between the sensor and the bottom of the lake.
The sensor used in these tests was a frustum sensor, with a lateral field of view (FOV) of 2.27 *rad*, and the onward FOV of 0.52 *rad*. Robot's optimal turning radius was configured to be $1m$, whereas the sampling distances were equal to $0.5m$ for segments and $0.1m$ for arcs respectively.
Performance comparison was carried out between smoothed (Fig. 4.2b) and unsmoothed (Fig. 4.2a) trajectories. Heatmaps depicted in Fig. 4.2c,4.2d represent the expected coverage of the area (along with the path), based on the generated trajectory. For these scenarios, there is no coverage loss in both cases.
The robot's operation was tested in water, with the resulting robot's trajectories depicted in Fig. 4.3.
Because of a high yaw-related gain and smaller number of waypoints, turning in

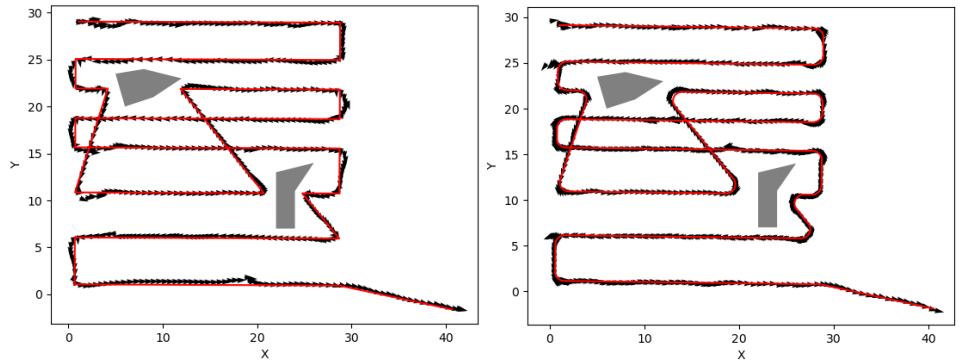


(a) Unsmoothed path – trajectory. (b) Smoothed path – trajectory.



(c) Unsmoothed path – heatmap. (d) Smoothed path – heatmap.

Figure 4.2: Trajectories and heatmaps generated for smoothed and unsmoothed scenario.



(a) Unsmoothed trajectory. (b) Smoothed trajectory.

Figure 4.3: Trajectories from experiments. Black arrows correspond to the actual robot poses, while the red lines mark the desired trajectory.

metric	smoothed	unsmoothed
traversing time	1529 s	1241 s
coverage loss	0%	0%
trajectory error	293.44 m	452.20 m

Table 4.1: Comparison of metrics chosen for evaluation of the smoothed and unsmoothed trajectories.

	1st	5th	10th	25th	50th	75th	90th	95th	99th
unsmoothed	0.01	0.03	0.04	0.07	0.13	0.23	0.34	0.47	0.63
smoothed	0.01	0.02	0.03	0.05	0.08	0.13	0.21	0.28	0.54

Table 4.2: Percentile's values for smoothed and unsmoothed trajectories errors (in meters).

place turned out to be more efficient in terms of time than following a smoothed trajectory. This observation is supported by the data shown in Tab. 4.1. Despite this, the scenario involving the smoothed trajectory demonstrated superior trajectory-following properties.
In evaluating the accumulated trajectory error—which is represented by the Euclidean distance between the ground truth and the actual trajectory—the smoothed path yields a 35% reduction compared to the segment-based path. Furthermore, the error distribution associated with the smoothed trajectory is comparatively more desirable. As depicted in Tab. 4.2, 99% of errors are below 0.54 m in the smoothed scenario, as opposed to 0.63 m in the unsmoothed version. Although the smoothed version manifests more outliers, these errors are closely clustered around a lower mean value in comparison to the unsmoothed version, as demonstrated in Fig. 4.5. It is noteworthy that in the smoothed scenario, outliers in the x-axis exhibit much higher values compared to those along the y-axis.
The refined coverage path planning can be used for object search and environment mapping – the sensor's measurements after traversing the whole trajectory are depicted in a form of an lake's bottom scan in Fig. 4.4.

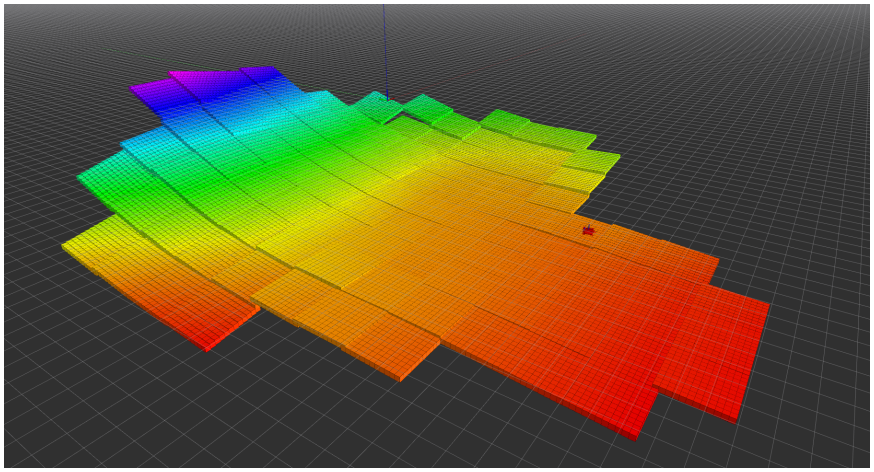
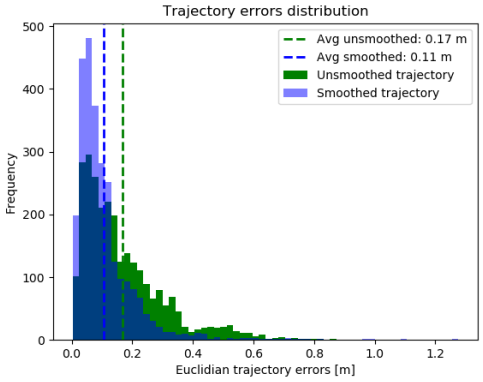
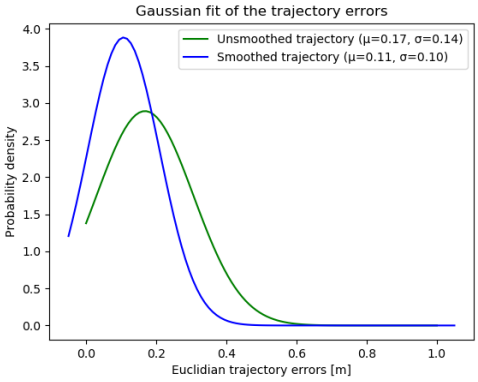


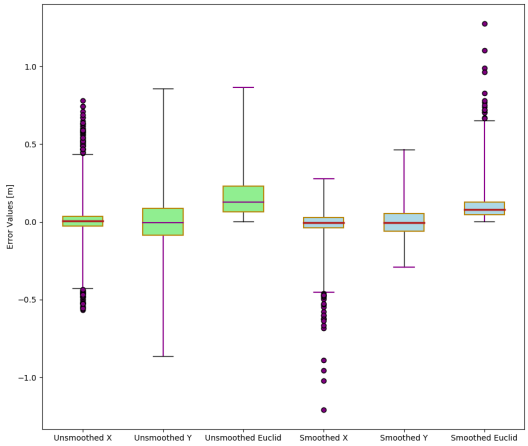
Figure 4.4: The mapped bottom of the lake.



(a) Histogram of the trajectory errors.



(b) Gaussian curve fits into the errors.



(c) Trajectory errors with respect to x, y and Euclidean measure for both trajectories.

Figure 4.5: Trajectories error statistics.

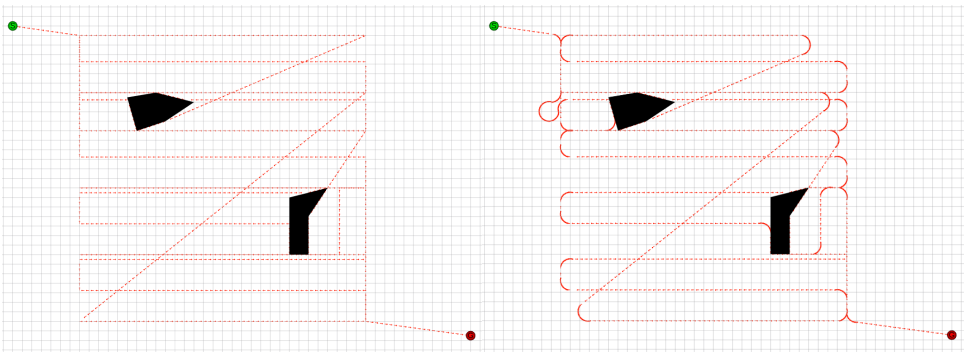
In light of the unexpected results where the robot demonstrated a faster turn-in-place than following a smoothed trajectory, a subsequent experiment was initiated to investigate these dependencies further. The updated experiment employed a smaller map with dimensions of 7.5m×7.5m, and incorporated a larger number of turns (inclusive of the outer smoothing scenario). The yaw-related gain in the controller was reduced to mimic the behavior of a robot with distinct kinematic properties. Furthermore, an increased number of waypoints was generated for the in-place rotations. The outcomes, as presented in Tab. 4.3, exhibited a reverse trend in comparison to the initial findings. The traversal time for the smoothed trajectory manifested an improvement at the expense of significant degradation in the trajectory-following precision when contrasted with the unsmoothed scenario. This indicates the complex dependencies between controller gains, robot kinematics, and path characteristics in influencing overall performance.

Additionally, this study investigated a scenario that, despite not displaying a significant difference in coverage between smoothed and unsmoothed paths under the initial conditions, revealed a noticeable disparity when reducing the lateral overlap and the lateral FOV. Upon simulating the change of lateral overlap to 0% and modifying the lateral FOV, a consequential rise in disparity was observed. Specifically, the difference increased up to 1.9% for the smoothed scenario, whereas for the unsmoothed one, it remained negligibly low at 0.0001%. Representative trajecto-

metric	smoothed	unsmoothed
traversing time	531 s	703 s
coverage loss	0%	0%
trajectory error	283.98 m	239.34 m

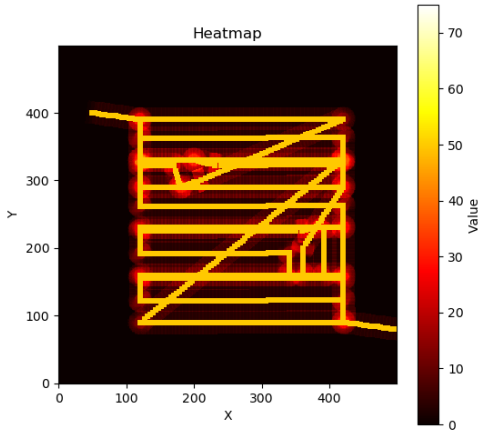
Table 4.3: Comparison of metrics chosen for evaluation of the smoothed and unsmoothed trajectories – decreased yaw-related gain.

ries under these conditions are illustrated in Fig. 4.6a,4.6b, and the corresponding heatmaps in Fig. 4.6c,4.6d. It can be observed that the coverage loss occurs with path segments forming acute angles in the inner smoothing scenario – the closer the angle is to 0^o, the further the curve is from the original waypoint, hence the higher the coverage loss is.

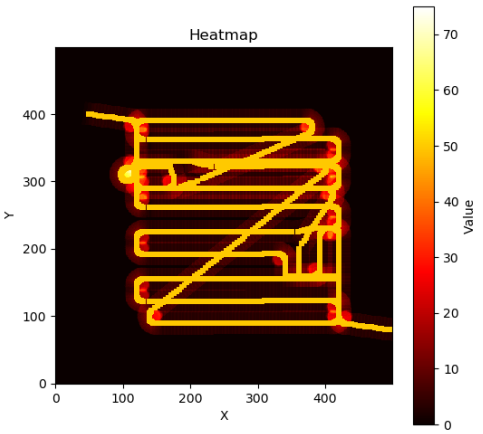


(a) Unsmoothed path – trajectory.

(b) Smoothed path – trajectory.



(c) Unsmoothed path – heatmap.



(d) Smoothed path – heatmap.

Figure 4.6: Trajectories and heatmaps corresponding to the coverage of smoothed and unsmoothed paths generated for an example showing the coverage difference.

Chapter 5

Conclusions

This project introduces an improvement in the domain of coverage path planning. It emphasizes post-processing measures, such as path smoothing and trajectory generation. The resulting trajectory, where feasible, adopts the structure of Dubins path. The application of either the inner or outer smoothing algorithm formulated as a constrained optimization problem depends on the positioning of the obstacles, with the latter presenting two distinct variants.

While the proposed smoothing algorithm already demonstrates satisfactory results, there exist room for further advancements. One such direction involves the integration of other successful methods like Bezier curves or clothoid-based smoothing. For instance, the application of these methods in inner smoothing combined with the use of Dubins curves for outer smoothing could potentially yield better results. Another potential area of exploration involves an investigation into the application of variable circle radii within the context of Dubins path. This adjustment could expand the applicability of smoothing across all cases, especially those involving closely-spaced waypoints.

The current algorithm implements a localized optimization strategy for path smoothing – specifically, if the smoothing operation cannot be initiated from the terminal point of the prior smoothing step, the path remains unsmoothed. A potential enhancement to this approach could involve the examination of global optimization strategies. By employing such methods, the number of turns subject to smoothing may be increased, thereby optimizing the path on a more comprehensive scale. This could yield improvements in the overall trajectory planning and enhance the performance of the navigation algorithm.

This project also attempted to implement a dynamic replanner. Despite the limitations posed by the three proposed approaches, there remain opportunities for extension of these methods. One such opportunity could involve the development of an algorithm that simplifies or approximates the polygon generated by sensor footprint rectangles. This step could considerably facilitate the decomposition process performed by the coverage path planner, making it more manageable and efficient.

In terms of practical implementation, a comparative analysis was conducted between unsmoothed and smoothed trajectories using real-world underwater robot testing. This comparison not only validated the efficacy of the proposed solutions but also provided valuable insights for future developments. While the unsmoothed paths demonstrated superior speed in traversing, the smoothed methodology manifested an enhancement in trajectory tracking precision up to 35% in comparison to the original pathway. Interestingly, this dynamic was inverted when the yaw-related gain in the controller was diminished and additional waypoints were incorporated in the in-place turns. Under these conditions, the segment-based trajectory had the improved tracking characteristics, however at the cost of extended traversal time.

Future investigations could concentrate on determining the correlations between these properties, broadening the scope of analyzed scenarios, and conducting experimental trials with robots exhibiting a diverse range of kinematic attributes in a underwater environment with stronger water currents.
In conclusion, this project proposes constrained optimization algorithms to smooth straight-line coverage paths using Dubins curves, thereby boosting the robot's trajectory tracking capabilities, with potential scope for enhancing traversal time efficiency. The project provides a foundation for further research in the field of smooth coverage path planning. Future work should focus on refining the introduced approaches, exploring and combining new methods for path smoothing, and developing more efficient algorithms for dynamic replanning.

Bibliography

[1] J. Trevelyan, S. Kang, and W. Hamel, *Robotics in Hazardous Applications*, 01 2008, pp. 1101–1126.

[2] Y. Petillot, G. Antonelli, G. Casalino, and F. Ferreira, "Underwater robots: From remotely operated vehicles to intervention autonomous underwater vehicles," *IEEE Robotics and Automation Magazine*, vol. PP, pp. 1–1, 05 2019.

[3] N. Barrett, J. Seiler, T. Anderson, S. Williams, S. Nichol, and S. Nicole Hill, "Autonomous underwater vehicle (auv) for mapping marine biodiversity in coastal and shelf waters: Implications for marine management," in *OCEANS'10 IEEE SYDNEY*, 2010, pp. 1–6.

[4] R. Wynn, V. Huvenne, T. Le Bas, B. Murton, D. Connelly, B. Bett, H. Ruhl, K. Morris, J. Peakall, D. Parsons, E. Sumner, S. Darby, R. Dorrell, and J. Hunt, "Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience," *Marine Geology*, vol. 352, 06 2014.

[5] D. Mindell and B. Bingham, "New archaeological uses of autonomous underwater vehicles," in *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295)*, vol. 1, 2001, pp. 555–558 vol.1.

[6] J. Kim, T. Kim, S. Song, M. Sung, and S.-C. Yu, "Parent-child underwater robot-based manipulation system for underwater structure maintenance," *Control Engineering Practice*, vol. 134, p. 105459, 2023.

[7] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, vol. 9, pp. 119 310–119 342, 2021.

[8] R. S. Pazmiño, C. E. Garcia Cena, C. A. Arocha, and R. A. Santonja, "Experiences and results from designing and developing a 6 dof underwater parallel robot," *Robotics and Autonomous Systems*, vol. 59, no. 2, pp. 101–112, 2011.

[9] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors*, vol. 18, no. 9, 2018.

[10] R. Bähnemann, N. Lawrance, J. J. Chung, M. Pantic, R. Siegwart, and J. Nieto, "Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem," pp. 277–290, 2021.

[11] R. Carvalho, H. Vidal, P. Vieira, and I. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," 04 1998.

[12] Y. Liu, W. Zhao, H. Liu, Y. Wang, and X. Yue, "Coverage path planning for robotic quality inspection with control on measurement uncertainty," *CoRR*, vol. abs/2201.04310, 2022.

[13] N. Basilico and S. Carpin, "Deploying teams of heterogeneous uavs in cooperative two-level surveillance missions," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 610–615.

[14] I. A. Hameed, "Coverage path planning software for autonomous robotic lawn mower using dubins' curve," in *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2017, pp. 517–522.

[15] N. Gyagenda, A. K. Nasir, H. Roth, and V. Zhmud, "Coverage path planning for large-scale aerial mapping," in *Towards Autonomous Robotic Systems*, K. Althoefer, J. Konstantinova, and K. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 251–262.

[16] C. Cai, J. Chen, Q. Yan, and F. Liu, "A multi-robot coverage path planning method for maritime search and rescue using multiple auvs," *Remote Sensing*, vol. 15, no. 1, 2023.

[17] M. Kloetzer and N. Ghita, "Software tool for constructing cell decompositions," in *2011 IEEE International Conference on Automation Science and Engineering*, 2011, pp. 507–512.

[18] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*, A. Zelinsky, Ed. London: Springer London, 1998, pp. 203–209.

[19] Y.-S. Jiao, X.-M. Wang, H. Chen, and Y. Li, "Research on the coverage path planning of uavs for polygon areas," in *2010 5th IEEE Conference on Industrial Electronics and Applications*, 2010, pp. 1467–1472.

[20] J. I. Vasquez Gomez, M. M. Melchor, and J. C. Herrera Lozada, "Optimal coverage path planning based on the rotating calipers algorithm," in *2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*, 2017, pp. 140–144.

[21] J. Huang, W. fu, S. Luo, C. Wang, B. Zhang, and Y. Bai, "A practical interlacing-based coverage path planning method for fixed-wing uav photogrammetry in convex polygon regions," *Aerospace*, vol. 9, p. 521, 09 2022.

[22] A. Zelinsky, R. Jarvis, J. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," 2007.

[23] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 2, 2001, pp. 1927–1933 vol.2.

[24] B. Pang, Y. Song, C. Zhang, and R. Yang, "Effect of random walk methods on searching efficiency in swarm robots for area exploration," *Applied Intelligence*, vol. 51, no. 7, pp. 5189–5199, 2021.

[25] C. Li, Y. Song, F. Wang, Z. Wang, and Y. Li, "A bounded strategy of the mobile robot coverage path planning based on lorenz chaotic system," *International Journal of Advanced Robotic Systems*, vol. 13, no. 3, p. 107, 2016.

[26] G. Hao, Q. Lv, Z. Huang, H. Zhao, and W. Chen, "Uav path planning based on improved artificial potential field method," *Aerospace*, vol. 10, no. 6, 2023.

[27] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[28] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3d exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1462–1468.

[29] R. Tarjan, "Depth-first search and linear graph algorithms," in *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, 1971, pp. 114–121.

[30] J. Holdsworth, "The nature of breadth-first search," 02 1999.

[31] A. Javaid, "Understanding dijkstra algorithm," *SSRN Electronic Journal*, 01 2013.

[32] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A systematic literature review of a* pathfinding," *Procedia Computer Science*, vol. 179, pp. 507–514, 2021, 5th International Conference on Computer Science and Computational Intelligence 2020.

[33] Z. Wang and Z. Bo, "Coverage path planning for mobile robot based on genetic algorithm," in *2014 IEEE Workshop on Electronics, Computer and Applications*, 2014, pp. 732–735.

[34] Z. Chibin, W. Xingsong, and D. Yong, "Complete coverage path planning based on ant colony algorithm," in *2008 15th International Conference on Mechatronics and Machine Vision in Practice*, 2008, pp. 357–361.

[35] A. Lakshmanan, R. E. Mohan, B. Ramalingam, L. Anh Vu, P. Veerajagadeshwar, K. Tiwari, and M. Ilyas, "Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot," *Automation in Construction*, vol. 112, p. 103078, 04 2020.

[36] S. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 718–724, 2004.

[37] E. Galceran, R. Campos, N. Palomeras, M. Carreras, and P. Ridao, "Coverage path planning with realtime replanning for inspection of 3d underwater structures," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6586–6591.

[38] M. Coombes, T. Fletcher, W.-H. Chen, and C. Liu, "Optimal polygon decomposition for uav survey coverage path planning in wind," *Sensors*, vol. 18, p. 2132, 07 2018.

[39] G. Parlangeli and G. Indiveri, "Dubins inspired 2d smooth paths with bounded curvature and curvature derivative." *IFAC Proceedings Volumes*, vol. 43, no. 16, pp. 252–257, 2010, 7th IFAC Symposium on Intelligent Autonomous Vehicles.

[40] S. Bochkarev and S. L. Smith, "On minimizing turns in robot coverage path planning," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 1237–1242.

[41] A. Šelek, M. Seder, M. Brezak, and I. Petrović, "Smooth complete coverage trajectory planning algorithm for a nonholonomic robot," *Sensors*, vol. 22, no. 23, 2022.

[42] M. Brezak and I. Petrović, "Path smoothing using clothoids for differential drive mobile robots," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1133–1138, 2011, 18th IFAC World Congress.

[43] F. Zhou, B. Song, and G. Tian, "Bézier curve based smooth path planning for mobile robot," *Journal of Information and Computational Science*, vol. 8, pp. 2441–2450, 12 2011.

[44] B. Song, G. Tian, and F. Zhou, "A comparison study on path smoothing algorithms for laser robot navigated mobile robot path planning in intelligent space," vol. 7, pp. 2943–2950, 12 2010.

[45] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1071–1078.

[46] M. Nykolaychuk and F. Ortmeier, "Coverage path re-planning for processing faults," in *Intelligent Robotics and Applications*, H. Liu, N. Kubota, X. Zhu, R. Dillmann, and D. Zhou, Eds. Cham: Springer International Publishing, 2015, pp. 358–368.