SEPTEMBER 2025 M.Sc. in Aeronautics and Aerospace Engineering SİBEL TUTAR

REPUBLIC OF TÜRKİYE

GAZİANTEP UNIVERSITY

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES

COORDINATED AND AUTONOMOUS MOVEMENT OF UNMANNED GROUND AND AERIAL VEHICLES AT INDOOR AND OUTDOOR ENVIRONMENTS

M.Sc. THESIS

IN

AERONAUTICS AND AEROSPACE ENGINEERING

BY

SİBEL TUTAR

SEPTEMBER 2025

COORDINATED AND AUTONOMOUS MOVEMENT OF UNMANNED GROUND AND AERIAL VEHICLES AT INDOOR AND OUTDOOR ENVIRONMENTS

M.Sc. Thesis

in

Aeronautics and Aerospace Engineering

Gaziantep University

Supervisor

Asst. Prof. Dr. M. Orkun ÖĞÜCÜ

by

Sibel TUTAR

September 2025

COORDINATED AND AUTONOMOUS MOVEMENT OF UNMANNED GROUND AND AERIAL VEHICLES AT INDOOR AND OUTDOOR ENVIRONMENTS

submitted by **Sibel TUTAR** in partial fulfillment of the requirements for the degree of Master of Science in **Aeronautics and Aerospace Engineering, Gaziantep University** is approved by,

Prof. Dr. Çiğdem AYKAÇ
Director of the Graduate School of Natural and Applied Sciences …….....

Prof. Dr. Eyüp YETER
Head of the Department of Aeronautics and Aerospace Engineering …….....

Asst. Prof. Dr. M. Orkun ÖĞÜCÜ
Supervisor, Aeronautics and Aerospace Engineering
Gaziantep University …….....

Exam Date: 11 September 2025

Examining Committee Members:

Asst. Prof. Dr. Derya HAYDARGİL
Hasan Kalyoncu University …….....

Asst. Prof. Dr. Sohayb ABDULKERİM
Gaziantep University …….....

Asst. Prof. Dr. M. Orkun ÖĞÜCÜ
Gaziantep University …….....

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Sibel TUTAR

ABSTRACT

COORDINATED AND AUTONOMOUS MOVEMENT OF UNMANNED GROUND AND AERIAL VEHICLES AT INDOOR AND OUTDOOR ENVIRONMENTS

TUTAR, Sibel
M.Sc. in Aeronautics and Aerospace Engineering
Supervisor: Asst. Prof. Dr. M. Orkun ÖĞÜCÜ
September 2025
56 pages

In this study, we present autonomous exploration in unmapped environments and aim to achieve complete topological coverage without relying on prior maps. Many existing approaches assume pre-built environmental models or target sensor coverage rather than topology, so they fall short of this goal. We propose two Generalized Voronoi Diagram (GVD)–based strategies that achieve coverage by systematically visiting Voronoi vertices and traversing edges. The first uses a range-sensing ground robot and maintains equidistance from nearby obstacles to navigate precisely. The second forms a cooperative UAV–UGV team: a camera-equipped aerial vehicle guides the ground platform and enables predictive decisions. We implement both strategies in a unified co-simulation pipeline that integrates MATLAB/Simulink with Gazebo and includes realistic vehicle dynamics and sensor models. We evaluated the methods in four environments with topological complexity increases from 8 to 39 Voronoi vertices. The results revealed a clear trade-off between spatial accuracy and operational efficiency. The range-sensor method spatial achieved 3-6 times higher precision through geometric vertex computation. By contrast, the aerial-camera method completed missions two to three times faster and improved path efficiency by 17–21% through predictive navigation.

Key Words: Autonomous exploration, Unknown environments, Topological mapping, Generalized voronoi diagram, UAV-UGV cooperation, Co-simulation.

ÖZET

İNSANSIZ BİR KARA ARACININ VE İNSANSIZ BİR HAVA ARACININ AÇIK VE KAPALI ORTAMLARDA KOORDİNELİ VE OTONOM HARAKETİ

TUTAR, Sibel
Yüksek Lisans Tezi, Havacılık ve Uzay Mühendisliği
Danışman: Dr. Öğr. Üyesi M. Orkun ÖĞÜCÜ
Eylül 2025
56 sayfa

Bu tezde, insansız otonom araçlar kullanarak, bilinmeyen ortamlara ilişkin haritaların elde edilmesi amaçlanmıştır. İlgili literatürde olduğu gibi tam sensör kapsamasını hedefleyen uygulamaların aksine, bu çalışmada tam topolojik kapsama üzerine yoğunlaşılmıştır. Bu amaçla, Genelleştirilmiş Voronoi Diyagramlarına (GVD) dayanan iki farklı yaklaşım kullanılmıştır. Birinci yaklaşımda, otonom yer aracının, üzerine sabitlenen bir mesafe ölçer yardımıyla, en yakındaki iki objeye eşit mesafede kalmak suretiyle engeller arasında hareket etmesi sağlanmıştır. İkinci yaklaşımda ise, koordineli bir şekilde hareket eden otonom yer aracı ve hava aracı birlikte kullanılmıştır. Bu senaryoda, yer aracı üzerinde bulunan mesafe ölçer çıkarılmış, bunun yerine hava aracı üzerine bir kamera eklenmiştir. Çeşitli görüntü işleme teknikleri ile GVD elde edilmiş ve yer aracı için gerekli olan referans değerler hesaplanmıştır. Her iki senoryoda da, kontrolcü ve görüntü işleme araçlarının MATLAB/Simulink tarafında koşturulduğu, araç dinamiklerinin ve sensör modellerinin ise Gazebo tarafında koşturulduğu bir benzetim ortamı kullanılmıştır. Sonuçlar, çeşitli karmaşıklık düzeylerine sahip 4 farklı ortamda elde edilmiştir. Mesafe sensörü kullanılan birinci senaryoda, topolojik özelliklerin konumlarının 3-6 kat daha yüksek hassasiyet ile tespit edildiği görülmüştür. Kameranın kullanıldığı ikinci senaryoda ise keşif süresinin 2-3 kat daha kısa olduğu anlaşılmıştır.

Anahtar Kelimeler: Otonom keşif, Haritalandırılmamış ortam, Topolojik harita, Genellştirilmiş voronoi diyagramları, UAV-UGV koordineli çalışma, Ko-simülasyon

"Dedicated to my family"

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my esteemed advisor, Dr. M. Orkun ÖĞÜCÜ, for his knowledge, experience, and continuous support throughout every stage of this thesis. His guidance and patient assistance have been of great value to me during this study.

I am deeply grateful to my family, especially my mother, father, and siblings, for their unwavering support, patience, and love. Their presence and belief in me have been my greatest source of strength throughout this journey.

Finally, I sincerely thank all my friends with whom I have had the pleasure of working and everyone who has stood by me during this process.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF SYMBOLS

ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle
τ_ϕ	Roll torque applied to body
τ_θ	Pitch torque applied to body
τ_ψ	Yaw torque applied to body
ω_i	Angular velocities of rotors $i=1,2,3,4$
k_F	Thrust coefficient
k_M	Torque coefficient
l	Distance from rotor to center of mass
m	Total mass of the quadcopter
I_{xx} , I_{yy} , I_{zz}	Moments of inertia about the body x, y, and z axes
g	Gravitational acceleration

LIST OF ABBREVIATIONS

UGV	Unmanned Ground Vehicle
UAV	Unmanned Aerial Vehicle
VD	Voronoi Diagram
GVD	Generalized Voronoi Diagram
VV	Voronoi Vertex
CP	Corner Point
RS	Range Sensor-based Exploration
AC	Aerial Camera-based Exploration
RRT	Rapidly Exploring Random Trees
PRM	Probabilistic Roadmap
SLAM	Simultaneous Localization and Mapping
DWA	Dynamic Window Approach
TEB	Timed Elastic Bands
NBV	Next-Best-View Exploration Method
QP	Quadratic Programming
PI	Proportional + Integral Type Controller

CHAPTER 1
INTRODUCTION

1.1 Motivation

Autonomous mobile robots now operate across many domains to improve safety in hazardous settings, reach remote sites, execute critical tasks with high accuracy, and maintain efficient, continuous operation. Their deployments range from search-and-rescue in disaster zones [1] and construction under extreme conditions [2] to warehouse logistics in industrial facilities [3] and field work in agriculture [4]. The performance of these applications, however, is critically dependent on the capabilities of navigation algorithms. Effective systems avoid obstacles, remain robust to environmental uncertainty, adapt to diverse spatial layouts, and use computational and energy resources efficiently [5]. These demands, in turn, motivate distinct families of methods tailored to different operating conditions and objectives.

Classical navigation in unknown environments often adopts reactive, mapless strategies that convert sensor readings directly into actions. Examples include Bug algorithms [6], Artificial Potential Fields [7], the Vector Field Histogram [8], and the Follow-the-Gap method [9]. These methods run quickly and use little memory. Yet the absence of spatial memory means they do not record where the robot has been, so they frequently re-enter explored regions, accumulate redundant path segments, and waste distance and time. Lacking a global model, they also cannot compare candidate routes for optimality, which promotes detours and increases travel time. In short, mapless reactivity limits long-horizon reasoning and the execution of complex navigation tasks. In contrast to these reactive approaches, navigation algorithms can alternatively rely on explicit environmental maps to guide decision-making. When such a map is available, the navigation process can be separated into two distinct path planning and motion control phases. Planners such as Rapidly Exploring Random Trees (RRT) [10], Probabilistic Roadmaps (PRM) [11], and deep reinforcement learning–based planners [12] generate collision-free paths by exploiting the environment's representation. Motion planners including the Dynamic

Window Approach (DWA) [13], Timed Elastic Bands (TEB) [14], and Pure Pursuit [15] then track those paths while preserving dynamic feasibility. Regardless of the specific algorithms, map-based systems use look-ahead over the map to avoid dead ends and local minimas. Moreover, awaring the environmental structure enables the selection of higher-quality routes and more reliable overall navigation.

Given these advantages, the choice of map representation becomes a critical factor in determining overall system performance. Metric maps—occupancy grids or point clouds produced by Visual SLAM [16] or LiDAR SLAM [17]—encode geometry with high fidelity, which supports precise localization and fine-grained planning. Fidelity comes at a cost. Storage and computation scale with the number of cells (roughly quadratic in 2D and cubic in 3D), so dense maps strain resource-constrained platforms. In dynamic scenes, frequent updates further tax real-time planners operating on these rich structures. Topological maps generated by topological SLAM [18] take a different route: they compress the environment into a graph whose nodes mark distinctive places (e.g., corners, junctions) and whose edges mark traversable connections. This abstraction is cheap to store and enables fast routing by graph search. The trade-off is lost geometry—purely topological maps discard clearance information and therefore still require metric checks to guarantee collision-free motion. Generalized Voronoi Diagrams (GVDs) [19] bridge the gap. By encoding the medial axis of free space, GVD edges lie along maximal-clearance routes, embedding collision margins directly in the representation. Planners can therefore select safe paths without calling out to external metric validators. Because GVDs update locally, they also support incremental mapping in changing environments. In practice, this middle ground overcomes key limitations of both dense metric and purely topological maps, yielding a balanced, robust representation for geometrically complex spaces.

Map choice defines the spatial substrate, but exploration builds it. The central problem is to drive an autonomous agent through unknown space in a way that is safe, efficient, and coverage-oriented. Researchers have proposed several families of strategies, each with distinct trade-offs in coverage efficiency, computation, and safety. Frontier-based exploration [20] steers the robot toward the boundary between known and unknown regions. Under ideal sensing, it can guarantee complete

coverage. In practice, it can get trapped in concavities, and it typically ignores path cost and kinematic limits. Next-best-view (NBV) planning [21] chooses poses with high expected information gain and handles occlusions well in complex 3D scenes. The price is heavy online evaluation of viewpoints, and a greedy choice can yield globally suboptimal tours. Entropy-based schemes [22] target high-uncertainty areas in probabilistic maps, balancing exploration and exploitation. Frequent entropy updates over dense grids are costly, and sensor noise can skew the objective. Segment-based approaches [23] partition the space into rooms, corridors, or other semantic units; the resulting structure is human-interpretable and supports orderly coverage, but performance hinges on reliable segmentation and degrades in open or irregular layouts. Generalized Voronoi Diagram (GVD)–based exploration takes a different path. The map's topological skeleton supplies both targets and routes: Voronoi vertices naturally coincide with frontiers, so the planner selects goals without a separate frontier detector, and traversal proceeds along edges that maximize clearance from obstacles. As a result, collision avoidance is implicit rather than an external check. The graph structure also enables disciplined, repeatable coverage policies (e.g., visiting vertices and sweeping edges systematically). In dynamic scenes, local updates to the Voronoi structure incorporate new obstacles without full recomputation. Taken together, these properties yield a unified exploration framework whose goals, paths, and safety margins all arise from the same topological representation.

1.2 Related Work

A review of GVD-based navigation approaches in the literature reveals diverse applications and methodological contributions. A study given in [24] connects start and goal along Voronoi boundaries and then shortens the route using visibility-based shortcuts and Steiner-point insertion. Building on a similar idea, [25] constructs Voronoi roadmaps on occupancy grids, sparsifies and prunes the resulting graph, and computes smoothed shortest paths. Multi-robot settings appear in [26], where GVD roadmaps are constructed and then refined via shortcutting, while switching/yielding rules coordinate robots to keep motion collision-free. To accelerate planning, [27] filters GVD features into a tree and derives heuristic paths that guide motion planners such as Risk-RRT, cutting planning time and improving performance in trap-prone environments. A related guidance strategy in [28] organizes GVD features into

homotopy-consistent heuristic paths; treating nodes as sequential subgoals steers RRT sampling and enables real-time planning in trap spaces. Sampling is further refined by [29], which proposes GVDF-RRT*: a combination of GVD-informed sparse sampling, node direct connections, and a forward-optimization step that reduces initial samples, planning time, and path cost across multiple map types. Finally, [30] extracts collision-free shortest paths on an incrementally updated GVD, forms Voronoi corridors to shrink the search region, and performs state-lattice search with QP-based smoothing to obtain additional efficiency gains. Despite these advances, all of the above methods assume an a priori map and therefore do not address exploration in truly unknown environments.

Map generation optimization is tackled in [31] by incrementally constructing and repairing Generalized Voronoi Diagrams (GVDs) on occupancy grids and propagating local changes, yielding order-of-magnitude speedups for multi-robot path planning. To improve mapping quality, [32] couples feature-based RGB-D SLAM with frontier extraction and a quality-aware target selector, then routes with a modified Voronoi planner to preserve clearance; real-world tests report higher mapping accuracy and lower total path length and time. For exploration efficiency, [33] extracts a GVD online from SLAM maps using a training-free network with distance-pooling layers and a Laplacian operator, identifies heuristic frontiers from GVD node radii, and fuses them via a multi-strategy assignment, which reduces both time and path cost in simulation and field trials. Despite these gains, the emphasis remains on faster point-to-point navigation rather than systematic exploration with coverage guarantees.

Exploration is accelerated in [34] by constructing reduced, approximated Voronoi graphs on occupancy grids and using them to drive next-best-view selection; this strategy cuts overall exploration time and attains full sensor coverage in both simulation and a real museum deployment. Building on a different idea, [35] performs dynamic Voronoi partitioning and optimizes a multi-objective cost, while a deep-reinforcement-learning controller handles navigation; together they yield high sensory coverage with few collisions in unknown environments. A hierarchical scheme in [36] further combines Voronoi partitioning with a GRU-augmented MATD3 controller and a prior-guided adaptive hybrid reward, reporting over 90%

exploration coverage with shorter mission times and fewer redundant revisits. Despite these outcomes, the emphasis in all three studies is sensory completeness—maximizing the area observed by sensors—rather than topological completeness, which demands exhaustive traversal of all accessible regions.

1.3 Contributions

Building on the above review, three gaps emerge: many methods assume an a priori map, few offer true coverage guarantees, and most optimize sensory completeness rather than topological completeness. To close these gaps, we introduce two GVD-based exploration strategies for unknown environments that guarantee topological completeness by systematically visiting Voronoi vertices and traversing the connecting edges.

In the first strategy, a range-sensor ground robot constructs the GVD reactively via closest-point detection and geometric vertex computation; it navigates with high precision but follows a deliberately conservative policy. In the second, a cooperative UAV–UGV team is used: a gimbal-stabilized, downward-looking camera on the aerial vehicle provides predictive visual guidance to a ground robot that does not depend on a specific onboard range sensor, enabling priority-driven, strategic exploration. We evaluate both strategies on four layouts of increasing complexity and compare their trajectories to ground truth, revealing a clear trade-off between spatial accuracy and operational efficiency.

In general, the contributions of this study to the literature can be summarized as follows:

- First systematic comparison of GVD-based reactive vs predictive exploration
- Guaranteed complete topological coverage in unknown environments
- The study reveals critical trade-offs between accuracy and efficiency through comprehensive evaluation
- 3-6-fold superior spatial precision of range sensor through geometric computation
- 2-3-fold exploration time improvements of aerial camera via predictive navigation

The remainder of this thesis proceeds as follows: Section 2 summarizes Voronoi and generalized Voronoi diagram fundamentals; Section 3 details the range-sensor strategy and its implementation; Section 4 develops the aerial-camera cooperative method and vision pipeline; Section 5 provides comparative simulation results across multiple environmental configurations; Section 6 discusses the fundamental trade-offs and practical implications; and Section 7 concludes with contributions and directions for future work.

CHAPTER 2
BASIC CONCEPTS

We first review Voronoi-based spatial representations and the mathematics that support them. The discussion starts with the classical Voronoi diagram; its standard terminology, its core properties, and the associated geometric interpretations. We then extend this framework to Generalized Voronoi Diagrams (GVDs), which treat finite-sized sites rather than idealized points. This extension matters for robotics: obstacles and environmental features occupy space, and their extent must be modeled. Throughout, we assume that the geometric attributes of the objects and the environment are known in advance, and we develop the concepts under that assumption.

2.1 Voronoi Diagram

Voronoi diagrams partition a domain into cells, each cell containing the points closer to one reference site than to any other. This construction encodes distance and adjacency relations that matter across computational geometry and its applications. Because the partition follows directly from a metric, the structure represents clean properties and broad practical value. Before developing results and algorithms, we set out the core components and terminology of the diagram:

- **Site (or Seed):** A point $p_i \in R^n$ from a finite set $S = \{p_1, p_2, ..., p_k\}$. Each site generates a corresponding region in the space.
- **Voronoi Region (or Voronoi Cell):** For each site p_i, the Voronoi region $V(p_i)$ is the set of all points in R^n that are at least as close to p_i as to any other site in S.
- **Voronoi Edge:** The boundary shared between two adjacent Voronoi regions. It consists of all points that are equidistant to two sites and closer to them than to any other site in S.

- **Voronoi Vertex:** A point that is equidistant to three or more sites. These vertices lie at the intersection of multiple Voronoi edges and represent points of higher-order symmetry.
- **Voronoi Diagram:** The collection of all Voronoi regions corresponding to the sites in S, denoted $\mathcal{V}(S) = \{V(p_1), V(p_2), \ldots, V(p_k)\}$. It forms a complete, non-overlapping partition of the space based on Euclidean proximity.

These concepts are illustrated in Figure 2.1, which shows a Voronoi diagram constructed from four distinct sites.

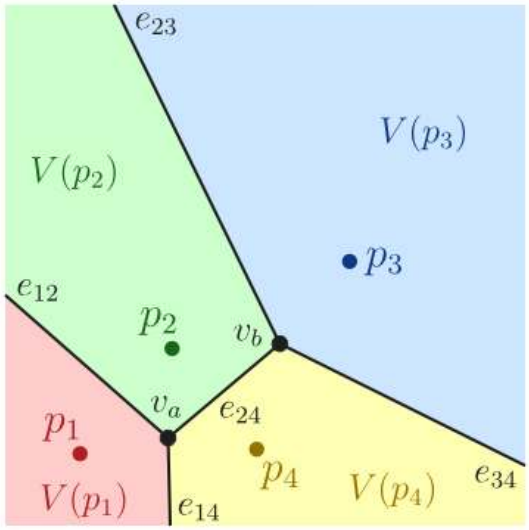


Figure 2.1 Illustration of a Voronoi diagram in R^2, showing key components: sites (p_1, p_2, p_3 and p_4), Voronoi regions ($V(p_1)$, $V(p_2)$, $V(p_3)$, and $V(p_4)$), Voronoi edges (e_{12}, e_{14}, e_{23}, e_{24}, and e_{34}), and Voronoi vertices (v_a and v_b).

The mathematical description of the Voronoi diagram is formally presented in Definition 1, specifying the partition of space into regions associated with the nearest site.

Definition 1: Let $S = \{p_1, p_2, ..., p_k\} \subset R^n$ be a finite set of k distinct points, called *sites*. The Voronoi diagram of S, denoted by $\mathcal{V}(S)$ is the partition of $\mathbb{R}^n$ into k regions $V(p_i)$ one for each site p_i, defined by

$$V(p_i) = \left\{ x \in R^n : \|x - p_i\| \leq \|x - p_j\| \ \forall j \neq i \right\}$$

where, $\| \cdot \|$ denotes the standard Euclidean norm. Here, each region $V(p_i)$ is the Voronoi cell corresponding to site p_i.

This spatial partitioning is exemplified in Figure 2.2, which depicts how the Voronoi diagram maintains clarity and interpretability even as the number of sites increases. The diagram enables immediate visual identification of the closest site to any selected point in the plane, without the need for explicit distance calculations.

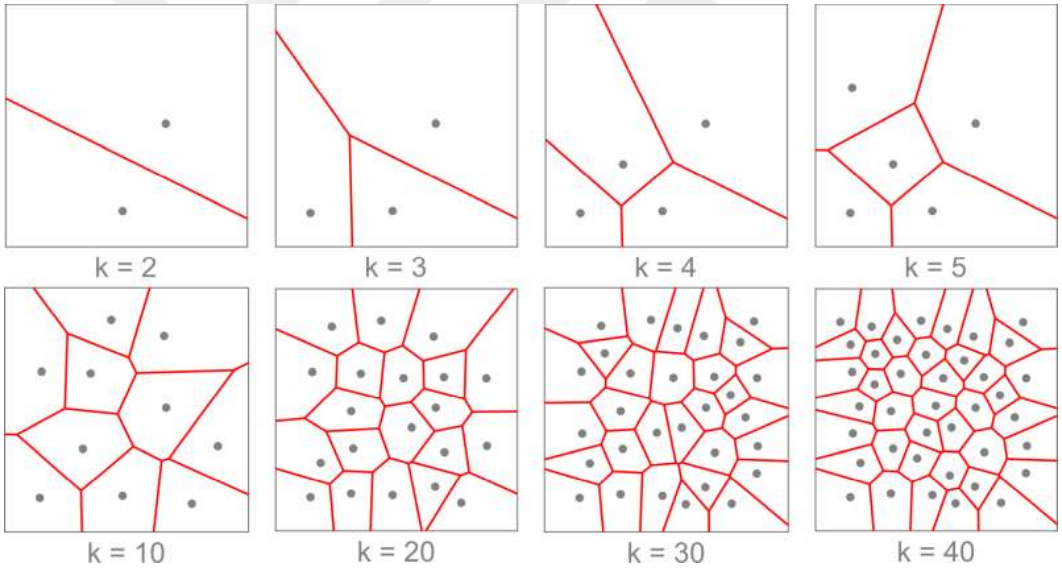


Figure 2.2 Voronoi diagrams in R^2 generated for various numbers of sites. Each diagram partitions the space into regions where every point is closest to a specific site. Here, the solid lines represent Voronoi edges, the dots represent sites, and the parameter k indicates the number of sites in each diagram.

Researchers construct Voronoi diagrams with several algorithmic families—brute-force evaluation, divide-and-conquer, Fortune's sweep line, incremental insertion,

and raster-based approximations, each trading off complexity, scalability, and optimality [37, 38]. Because the structure partitions space purely by distance, it underpins a wide range of scientific and engineering workflows. In computational science, weighted Voronoi partitions dynamically distribute domains across processors to balance load even on nonuniform meshes [39]. Telecommunications planners apply Voronoi-based decompositions to multi-tier 5G radio access networks, improving base-station placement while raising spectral and energy efficiency [40]. Transportation studies use Voronoi regions to define service zones around railway stations and to enable hybrid public-transport planning that respects spatial and temporal proximity [41]. In autonomous navigation, threat-aware UAV planners route vehicles along maximal-clearance Voronoi corridors to improve survivability in dynamic environments [42].

2.2 Generalized Voronoi Diagram

The classical Voronoi diagram treats each site as a dimensionless point. That abstraction works well at large scales. For example, in long-range planning where obstacles or landmarks can be approximated as point references. When the robot's size becomes comparable to the surrounding objects, however, the point-site assumption breaks down. The formulation is then generalized by modeling sites as spatially bounded shapes rather than points; the resulting structure is the Generalized Voronoi Diagram (GVD). The formal construction appears in Definition 2, which extends the classical diagram to accommodate non-point sites with finite extent.

Definition 2: Let $\mathcal{O} = \{O_1, O_2, \ldots, O_k\}$ be a finite set of mutually disjoint, closed objects in R^n, where each $O_i \subset R^n$ may be a point, curve, polygon, or other bounded geometric shape. The Generalized Voronoi Diagram (GVD) of $\mathcal{O}$, denoted $\mathcal{V}(\mathcal{O})$, is a partition of the domain into k regions $V(O_i)$, where each region is defined as:

$$V(O_i) = \left\{ x \in R^n : d(x, O_i) \leq d(x, O_j) \quad \forall\, j \neq i \right\}$$

where, $d(x, O_i) = \inf\{\|x - y\| : y \in O_i\}$ denotes the shortest Euclidean distance from point x to object O_i.

A bounded environment populated with primitive shapes - such as rectangles, circles, and stars - illustrates the structure of a Generalized Voronoi Diagram, where the drawn edge segments trace the set of locations that are equidistant to two surrounding objects, as shown in Figure 2.3.

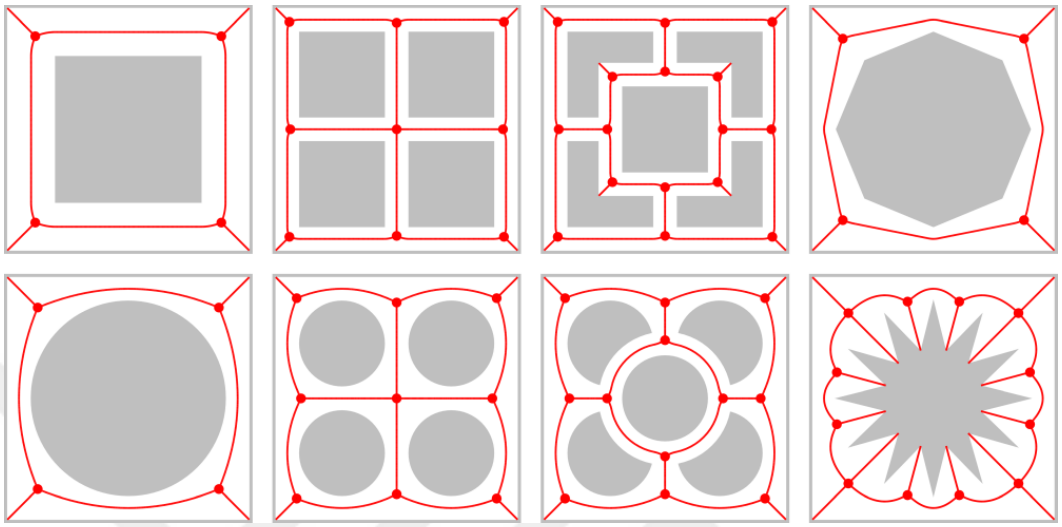


Figure 2.3 Generalized Voronoi Diagrams in R^2 computed for a bounded domain containing polygonal and curved obstacles. The diagram illustrates Voronoi edge segments that represent loci of points equidistant to the closest pair of surrounding objects, capturing the medial geometry of the environment. Moreover, intersection points of three or more edges, known as Voronoi vertices, are indicated with dots.

Computing Generalized Voronoi Diagrams (GVDs), which extend classical diagrams to finite-size sites such as curves, polygons, or composite shapes, remains as a challenging problem. Hence, practitioners choose methods based on resource budgets, target resolution, and how the environment is represented. Grid-based distance transforms (e.g., the Euclidean transform or brushfire) are widely adopted for their robustness and straightforward implementation; subsequent skeletonization recovers medial axes via morphological or geometric thinning. Wavefront propagation simulates expanding fronts from object boundaries and often suits robotic pipelines. When exact geometry is available, analytic constructions apply; when flexibility matters more than precision, sampling-based approximations suffice [43]. Despite the additional cost relative to point-site diagrams, GVDs support tasks

that depend on proximity among finite-sized objects. In medical image analysis, they drive cell segmentation that preserves topological adjacency among touching, irregular morphologies and improves fluorescence microscopy accuracy over conventional methods [44]. In stereo satellite processing, they encode local spatial relationships that help homography-based matchers distinguish true line-segment correspondences from parallel distractions [45]. In geographic knowledge discovery, they induce topological partitions from mixed spatial primitives and weights, enabling analyses from urban planning to emergency response [46]. In additive manufacturing, they decompose complex porous structures into manufacturable regions, yielding toolpaths with better material efficiency, continuity, and structural fidelity [47].

We review classical Voronoi diagrams for point sites and their generalized forms for finite-size objects, together with the underlying mathematics, construction methods, and application areas. Most of these formulations assume that the geometry of the environment and its objects is known in advance. In practice, especially in robotic exploration and autonomous navigation that assumption often fails. When the map is partial or absent, the robot must build it online by fusing sensor observations into a spatial representation. In this study, we develop incremental techniques that construct Generalized Voronoi Diagrams directly from sensor data. The following sections present this contribution in detail.

CHAPTER 3
EXPLORATION WITH AN AUTONOMOUS GROUND VEHICLE

We explore an unknown environment by deriving a topological connectivity map as a Generalized Voronoi Diagram (GVD), directly from range measurements. No prior information about object geometry or location is assumed. A differential-drive ground robot carries a range sensor that supplies local perception and supports online map construction. Section 3.1 summarizes the differential drive autonomous ground vehicle model. Section 3.2 outlines the co-simulation framework and system implementation. Section 3.3 explains important functional blocks within simulation framework.

3.1 Vehicle Model

A differential-drive mobile robot, Pioneer 2-DX, is used as a ground platform. Locomotion is produced by two independently actuated wheels mounted on a common axle. A third caster wheel provides support for balance; because it is unpowered, it contributes no drive torque and is omitted from the system model. The motion depends on the relative wheel speeds.

1. When both wheels rotate in the same direction at equal speeds, the robot travels in a straight line (forward or backward).
2. When the wheels rotate at equal speeds in opposite directions, the robot spins in place about the midpoint of the axle.
3. When the wheels rotate in the same direction but at different speeds, the robot follows a circular arc whose instantaneous center of curvature lies on the line colinear with the axle but outside the axle segment.

The third case is illustrated in Figure 3.1, and the associated parameters are summarized in Table 3.1. For the specified wheel speeds (w_l, w_r), the angular velocity of the body (W) and the radius of the resulting circular motion (R) can be calculated using the equations provided below.

$$w = \frac{r}{l}\left(w_r - w_l\right) \qquad R = \frac{l}{2}\frac{w_r + w_l}{w_r - w_l} \tag{3.1}$$

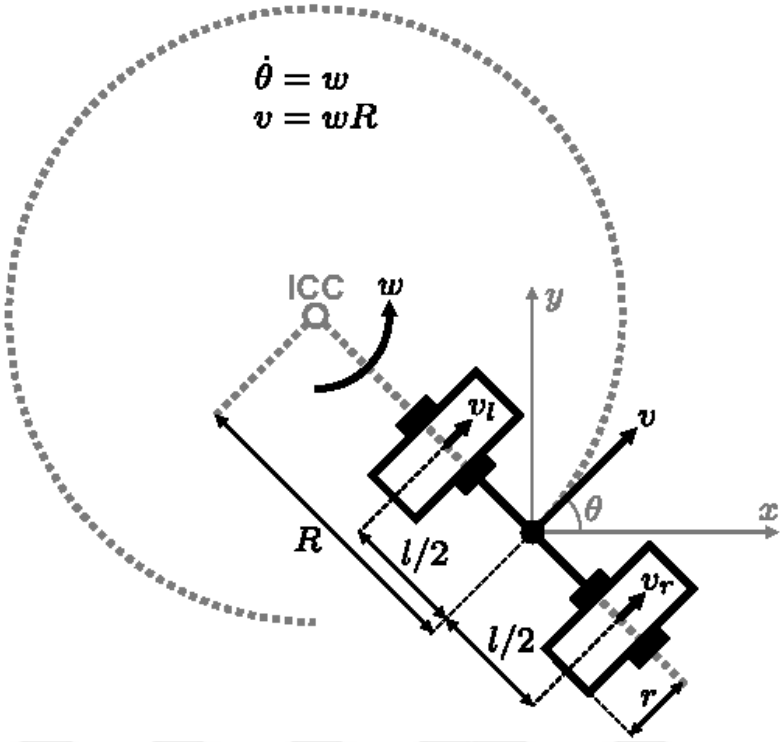


Figure 3.1 Geometric representation of differential drive vehicle performing circular motion with different wheel speeds

Table 3.1 Wheel speeds and corresponding vehicle motion parameters

Symbol	Description
w_r, w_l	Right and left wheels angular velocity
v_r, v_l	Right and left wheels linear velocity
x, y	Vehicle linear position in global frame
v	Vehicle linear velocity at axle midpoint
θ, W	Vehicle angular position and angular velocity
τ_r, τ_l	Right and left wheel input torques
F_r, F_l	Right and left wheels traction forces
r	Wheel radius
J_w	Wheel moment of inertia
l	Axle width (or wheelbase length)
m	Vehicle mass
I_z	Vehicle moment of inertia about z-axis
ICC	Instantaneous center of curvature
R	Radius of circular motion

The kinematic model describes the relationship between wheel velocities and vehicle motion without considering forces or torques. To obtain derivative of position, average wheel velocity is projected onto vehicle heading.

$$\dot{x} = \frac{v_r + v_l}{2}\cos(\theta) \qquad \dot{y} = \frac{v_r + v_l}{2}\sin(\theta) \tag{3.2}$$

To obtain angular velocity, difference between wheel velocity is divided by wheelbase.

$$\dot{\theta} = \frac{v_r - v_l}{l} \tag{3.3}$$

For control applications, the kinematic model can be expressed in matrix form. The state derivatives can be expressed as a function of wheel velocities through the transformation matrix shown below.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\cos(\theta) & \frac{1}{2}\cos(\theta) \\ \frac{1}{2}\sin(\theta) & \frac{1}{2}\sin(\theta) \\ \frac{1}{l} & -\frac{1}{l} \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix} \tag{3.4}$$

The dynamic model includes forces, torques, and inertial effects governing the motion. The right and left wheels angular acceleration are determined by subtracting the resistance from ground contacts and the applied torques.

$$J_w\dot{\omega}_r = \tau_r - rF_r \qquad J_w\dot{\omega}_l = \tau_l - rF_l \tag{3.5}$$

The longitudinal acceleration is calculated by subtracting the drag force in x-direction from the sum of wheel forces.

$$m\ddot{x}_c = F_r + F_l - F_{drag,x} \tag{3.6}$$

The lateral acceleration is determined only by the drag force since no wheel force acts in y-direction due to the non-slip condition.

$$m\ddot{y}_c = -F_{drag,y} \tag{3.7}$$

The angular acceleration is obtained by subtracting the rotational drag from the torque generated by differential wheel forces.

$$I_z\ddot{\theta} = (F_r - F_l)\frac{l}{2} - \tau_{drag} \tag{3.8}$$

Non-holonomic constraint ensures that the vehicle cannot move sideways, perpendicular to the wheel orientation.

$$\dot{x}_c \sin(\theta) - \dot{y}_c \cos(\theta) = 0 \tag{3.9}$$

The right and left wheels linear velocities are calculated by multiplying the angular velocities with the wheel radius, assuming no slip occurs.

$$v_r = \omega_r r \qquad v_l = \omega_l r \tag{3.10}$$

3.2 Simulation Setup

We implement the range-sensor exploration in a co-simulation setup that couples MATLAB/Simulink for algorithm development and control design with Gazebo for high-fidelity, physics-based simulation. The virtual environment supplies ground-truth state and sensor data and enforces repeatable test conditions that are hard to reproduce on hardware. Consequently, the combined setup provides an efficient and realistic platform for developing and evaluating the method. A bidirectional TCP link maintains synchronized execution between the two simulators: Gazebo streams vehicle state and LiDAR measurements to MATLAB/Simulink, and the controller returns computed torque commands in the opposite direction as shown in Figure 3.2.

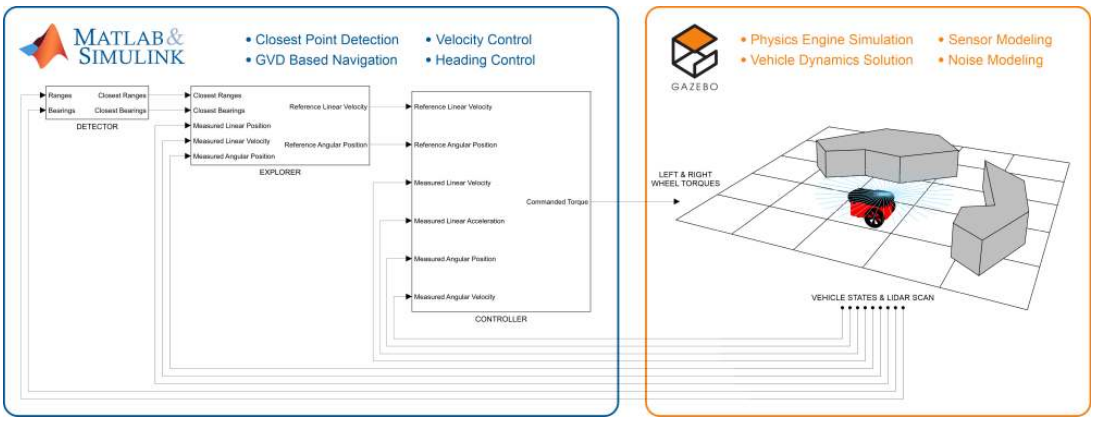


Figure 3.2 Co-simulation architecture integrating MATLAB/Simulink and Gazebo environments for LiDAR-based exploration

Gazebo hosts the full physics scene: a bounded 10 x 10 m arena populated with static obstacles of varied geometry. Inside this arena we deploy a Pioneer 2-DX differential-drive model with faithful kinematics and dynamics. The controller supplies individual wheel torques, and Gazebo's physics engine integrates these

inputs to yield the resulting translational and rotational motion. A Hokuyo LiDAR mounted on the robot provides full-circle perception: 360° scans with 1° angular spacing (−180° to +179°), a 10 m maximum range, and 0.01 m radial resolution. Zero-mean noise consistent with the sensor specification is injected to preserve simulation fidelity.

MATLAB/Simulink takes synchronized data streams from Gazebo. The data packet includes the vehicle state: linear position, velocity, and acceleration together with angular position and angular velocity, as well as LiDAR scans reported as range–bearing pairs. Within Simulink, algorithmic and control blocks filter and fuse these inputs, compute the required wheel torques. The commands are then returned to Gazebo over the TCP link, where they actuate the vehicle model and close the co-simulation loop.

3.3 Functional Blocks

This section examines three blocks: Detector, Explorer, and Controller which are implemented in MATLAB/Simulink as given in Figure 3.2. We start with the Detector, the front end of the perception pipeline. It takes raw LiDAR scans as range-bearing pairs and returns a filtered list of nearest surface contacts to the visible objects. Hence, the number of outputs matches the count of objects that are in line-of-sight from the robot's current pose. The core routine treats the polar range profile as a signal and searches for local minima. Each candidate minimum is then tested: does it correspond to a genuine nearest contact on a distinct object, or is it merely part of a larger geometric feature? For convex bodies, the procedure typically yields a single representative contact. For non-convex shapes, multiple contacts may be valid because different surface segments can lie at the same shortest distance. Curved boundaries need extra actions. In simulation, circles and smooth arcs are often faceted into short line segments, which can produce clusters of spurious minima along one physical surface. We suppress these artifacts in two steps. First, the module reconstructs ray-surface intersection locations from the measured ranges and bearings. Next, it evaluates the slope between adjacent intersections to assess surface continuity. When that slope falls below a preset threshold, indicating samples from the same continuous surface, the related samples are grouped and the shortest range within the group is retained as the representative measurement.

The second component, the Explorer block, implements a GVD-based navigation algorithm that systematically traverses the unknown workspace and covers all accessible regions. Using the filtered range–bearing measurements from the Detector, it issues linear velocity and heading (angular position) references that keep the vehicle at approximately equal clearance from nearby objects (along the medial axis). The pseudo-code belongs to this approach is listed in Algorithm 1.

Algorithm 1 Range Sensor-based GVD Exploration Algorithm

```
Initialize empty VertexList
Position robot on Voronoi edge by equalizing distances to two closest objects
while untraced paths exist do
    direction ← CalculateBisector(closest_obj_1, closest_obj_2)
    Generate velocity and orientation references for direction
    Move along direction until vertex detection or obstacle encounter
    if front distance ≤ threshold then
        Mark position as corner point and return to previous vertex
    else if three or more objects equidistant then
        current_vertex ← detected vertex position
        if current_vertex ∈ VertexList then
            Update path completion status for current and previous vertices
        else
            Add current_vertex to VertexList with path information
            Update path completion status for current and previous vertices
        end if
        if untraced paths exist at current_vertex then
            Select direction from available untraced paths
        else
            Navigate to nearest vertex with untraced paths
        end if
    end if
end while
return Reference commands for continued exploration
```

It maintains a vertex list that records junctions already visited together with per-edge completion flags. When the robot arrives at a vertex, where three or more objects are equidistant, the module keeps the location and evaluates the outgoing directions that remain unvisited. At each such junction, untraced edges are given priority and one is selected for continuation. If all edges incident to the current vertex have been traversed, the system backtracks to the nearest vertex that still contains unvisited edges. Repeating this select-and-backtrack cycle yields exhaustive coverage of the environment's reachable topology. Figure 3.3 shows how the robot explores the unknown area and how its path grows over time. Gray areas are fixed obstacles. Light-blue rays start at the robot and show the raw range readings. Dark-blue lines mark the subset to the nearest objects after the Detector filter. Solid red lines are

Voronoi edges the robot has already followed. Dashed pink lines show the parts of the diagram that are not yet traced. The numbered red circles (v_1, v_2, …, v_8) mark vertices that were visited and added to the vertex list earlier in the run. Corner points c_1 and c_2 mark detected obstacle corners. Together, these layers make it easy to see what the robot senses and where it has moved.

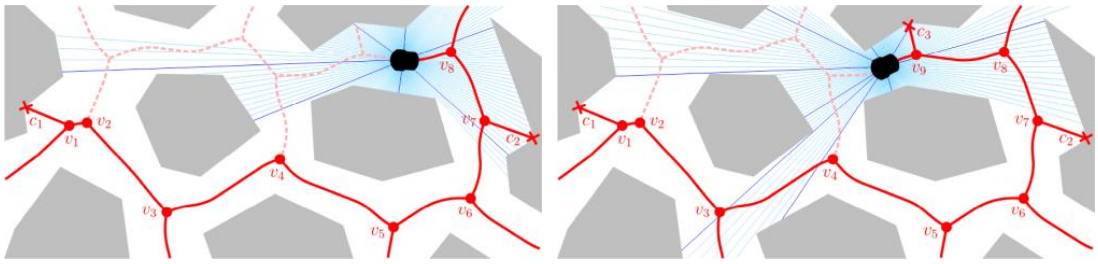


Figure 3.3 Sequential progression of the range sensor-based Voronoi exploration algorithm: a) before vertex detection (left) b) after vertex detection (right)

As shown on the left side, vertices v_1 through v_8 have already been visited and their outgoing paths have been traced. From its current pose, the robot continues along the traced edge and follows the bisector of the two nearest objects toward a yet-unexplored part of the map. As shown on the right side, the robot reaches a new junction and the algorithm detects a fresh vertex, labeled as v_9. It is not in the list, so the vertex list is updated. Three outgoing paths are visible at headings −170°, −10°, and +110°. The vehicle arrived via the −10° branch, so that branch is marked as traced. For the next move, the algorithm chooses the untraced branch at +110°. It proceeds along the bisector between two surfaces of the same non-convex object. It nears the corner where those surfaces meet. The forward range drops below the threshold, which triggers the corner detector and yields the corner point c_3. The robot then goes back to v_9. The status is updated: the −10° and +110° paths are now traced. Only the −170° branch remains. The algorithm selects it and the vehicle continues along the bisector of the two closest objects, keeping the same navigation behavior until it reaches the next vertex.

The third component, the Controller block, acts as the vehicle's actuation interface. As inputs, it takes the linear velocity and angular position references from Explorer

block, as well as linear velocity, linear acceleration, angular position and angular velocity measurements from Gazebo. Using these signals, it computes the right and left wheel torques which is fed through Gazebo for actuation. The control design follows a cascaded, two-loop layout with separate chains for linear and angular motion as shown in Figure 3.4. While the outer loops track the linear velocity and angular position setpoints, the inner loops regulate their derivates.

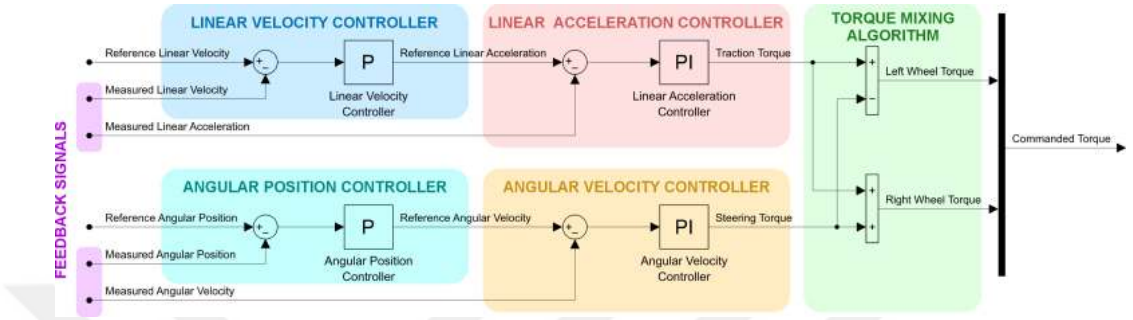


Figure 3.4 Controller block architecture showing cascaded double-loop control with separate linear and angular motion chains

For linear motion control, a proportional controller generates acceleration reference from velocity tracking error:

$$v_{err}(t) = v_{ref}(t) - v_{mea}(t) \quad \rightarrow \quad a_{ref}(t) = K_{p,v} \cdot v_{err}(t) \tag{3.11}$$

This acceleration reference is then processed by a proportional-integral controller to produce the traction torque:

$$a_{err}(t) = a_{ref}(t) - a_{mea}(t) \quad \rightarrow \quad \tau_{tra}(t) = K_{p,a} \cdot a_{err}(t) + K_{i,a} \int a_{err}(t)dt \tag{3.12}$$

For angular motion control, a proportional controller generates angular velocity reference from heading error:

$$\theta_{err}(t) = \theta_{ref}(t) - \theta_{mea}(t) \quad \rightarrow \quad \omega_{ref} = K_{p,\theta} \cdot \theta_{err}(t) \tag{3.13}$$

This is followed by a proportional-integral controller that produces the steering torque:

$$\omega_{err}(t) = \omega_{ref}(t) - \omega_{mea}(t) \quad \rightarrow \quad \tau_{ste}(t) = K_{p,\omega} \cdot \omega_{err}(t) + K_{i,\omega} \int \omega_{err}(t)dt \tag{3.14}$$

The mathematical symbols and their corresponding definitions used throughout the Equations 3.11 to 3.14 are summarized in Table 3.2.

Table 3.2 Controller equation symbols, variables, and gain parameters

Related Variable	Reference	Measured	Error	Proportional Gain	Integral Gain
Linear Velocity	$v_{ref}(t)$	$v_{mea}(t)$	$v_{err}(t)$	$K_{p,v}$	-
Linear Acceleration	$a_{ref}(t)$	$a_{mea}(t)$	$a_{err}(t)$	$K_{p,a}$	$K_{i,a}$
Angular Position	$\theta_{ref}(t)$	$\theta_{mea}(t)$	$\theta_{err}(t)$	$K_{p,\theta}$	-
Angular Velocity	$\omega_{ref}(t)$	$\omega_{mea}(t)$	$\omega_{err}(t)$	$K_{p,\omega}$	$K_{i,\omega}$

The traction torque component, $\tau_{tra}(t)$, drives forward or backward movement, while the steering torque component, $\tau_{ste}(t)$, creates the necessary speed differential between wheels to achieve the desired heading changes. The final stage employs a torque mixing algorithm that combines these torque components to generate individual wheel commands. This straightforward computational process distributes the motion requirements between the left and right wheels through simple addition and subtraction operations:

$$\tau_{left}(t) = \tau_{tra}(t) - \tau_{ste}(t) \qquad \tau_{right}(t) = \tau_{tra}(t) + \tau_{ste}(t) \tag{3.15}$$

CHAPTER 4
EXPLORATION WITH MULTI-VEHICLE COOPERATION

We now explore the unknown environment via Generalized Voronoi Diagrams (GVDs) but using another approach. While the goal is the same as previous case—cover all reachable regions and recover a topological connectivity map, the sensing modality is different. We use a cooperative system with a differential-drive ground robot and a quadrotor that carries a downward-facing camera on a three-axis gimbal. The aerial vehicle flies at a fixed height above the ground vehicle and supplies visual guidance. Image processing on these camera feeds drives exploration and builds the GVD as the vehicle moves. In this scenario the ground vehicle has no onboard sensors; it relies entirely on the external view from the aerial platform. Similarly, the presentation is split into three parts. Section 4.1 sumarizes the aerial vehicle model. Section 4.2 describes the co-simulation setup and the system implementation. Section 4.3 then details the functional blocks.

4.1 Vehicle Model

This section introduces the dynamic model of the quadcopter used in the cooperative exploration framework. The rotor configuration, motor numbering scheme, propeller rotation directions, and the coordinate frame definitions including both inertial-frame and body-frame representations are shown in Figure 4.1.

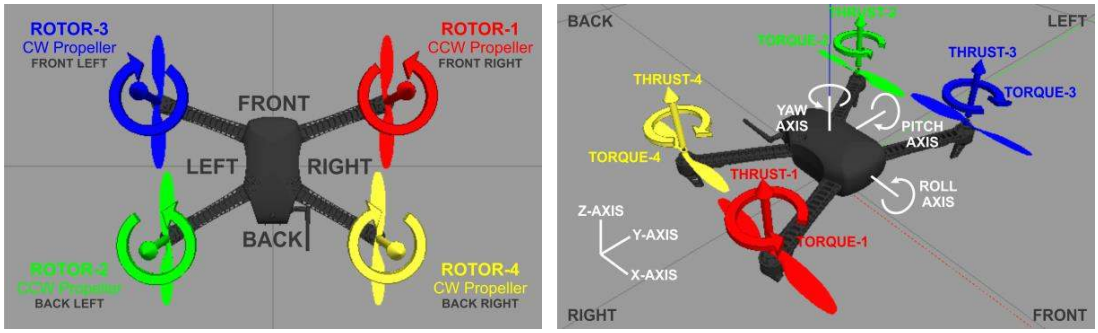


Figure 4.1 Quadcopter rotor configuration showing propeller rotation directions with body-frame and inertial-frame representations

Equations 4.1 – 4.3 describe the acceleration of the quadcopter's center of mass in the inertial frame according to the North-West-Up (NWU) convention. The acceleration in the North direction is determined by the projection of the total thrust vector onto the inertial x-axis.

$$m\ddot{x} = T(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi) \tag{4.1}$$

The acceleration in the West direction is determined by the projection of the total thrust vector onto the world y-axis.

$$m\ddot{y} = T(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi) \tag{4.2}$$

The acceleration in the Up direction is determined by the projection of the total thrust onto the world z-axis, counteracting the force of gravity.

$$m\ddot{z} = -mg + T(\cos\phi \cos\theta) \tag{4.3}$$

Additionally, Equations 4.4 – 4.6 describe the evolution of the body's angular velocities. The angular acceleration around the body x-axis is driven by the roll control torque and gyroscopic precession.

$$\dot{p} = \frac{1}{I_{xx}}\left[\tau_\phi + qr(I_{yy} - I_{zz})\right] \tag{4.4}$$

The angular acceleration around the body y-axis is driven by the pitch control torque and gyroscopic precession.

$$\dot{q} = \frac{1}{I_{yy}}\left[\tau_\theta + pr(I_{zz} - I_{xx})\right] \tag{4.5}$$

The angular acceleration around the body z-axis is driven by the yaw control torque and gyroscopic precession.

$$\dot{r} = \frac{1}{I_{zz}}\left[\tau_\psi + pq(I_{xx} - I_{yy})\right] \tag{4.6}$$

Moreover, Equations 4.7 – 4.9 relate the body angular rates (p , q , r) to the time derivatives of the Euler angles ($\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$). The rate of change of the roll angle is a function of the body rates and the current pitch and roll angles.

$$\dot{\phi} = p + q \sin\phi \tan\theta + r \cos\phi \tan\theta \tag{4.7}$$

The rate of change of the pitch angle is a function of the body rates and the current roll angle.

$$\dot{\theta} = q \cos\phi - r \sin\phi \tag{4.8}$$

The rate of change of the yaw angle is a function of the body rates and the current roll and pitch angles.

$$\dot{\psi} = \frac{q \sin\phi + r \cos\phi}{\cos\theta} \tag{4.9}$$

Finally, Equations 4.10 – 4.13 define the control inputs generated by the four motors. Total upward force acting on body is the sum of the thrust from all four propellers.

$$T = k_F(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \tag{4.10}$$

The moment around the body's x-axis is generated by the differential thrust between the pairs of rotors on the left-back/right-front and right-back/left-front axes.

$$\tau_\phi = \frac{l}{\sqrt{2}} k_F(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2) \tag{4.11}$$

The moment around the body's y-axis is generated by the differential thrust between the pairs of rotors on the left/right and front/back axes.

$$\tau_\theta = \frac{l}{\sqrt{2}} k_F(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \tag{4.12}$$

The moment around the body's z-axis is generated by the sum of the reaction torques from the clockwise and counter-clockwise rotating propellers.

$$\tau_\psi = k_M(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2) \tag{4.13}$$

Table 4.1 summarizes the complete set of variables and parameters used in the dynamic model of a quadcopter, including their physical descriptions.

Table 4.1 Nomenclature used in the nonlinear dynamic model of a quadcopter

Symbol	Description
x , y , z	Linear positions in the world frame (North, West, Up)
p , q , r	Angular velocities in the body frame
ϕ , θ , ψ	Euler angles (roll, pitch, yaw)
ω_1 , ω_2 , ω_3 , ω_4	Angular velocities of rotors 1 to 4
τ_ϕ , τ_θ , τ_ψ	Total torques applied to the body (roll, pitch, yaw)
T	Total thrust generated by all rotors
k_F	Thrust coefficient
k_M	Torque coefficient
l	Distance from rotor to center of mass
m	Total mass of the quadcopter
I_{xx} , I_{yy} , I_{zz}	Moments of inertia about the body x, y, and z axes
g	Gravitational acceleration

4.2 Simulation Setup

We implement the vision-based exploration in the same co-simulation setup as before: MATLAB/Simulink for algorithms and Gazebo for high-fidelity physics. Gazebo again hosts a 10 × 10 m bounded arena with fixed obstacles so the conditions match the range-sensor case. What changes here is the vehicle stack and the data link: a cooperative UAV–UGV pair requires a richer message set for images, states, and commands. The ground robot is a Pioneer 2-DX with accurate kinematic and dynamic models. The aerial platform is a 3DR Iris quadrotor modeled with realistic flight physics. We mount a three-axis gimbal on the UAV to stabilize a downward-facing camera. The gimbal keeps the camera steady and centered on the ground robot as exploration proceeds. The camera model is simple and explicit: 640×480 pixels, 60° horizontal and 47° vertical field of view, with focal lengths $f_x = 554$ and $f_y = 554$ pixels. At an altitude of about 4 m, the footprint on the ground is roughly 4.6 m × 3.5 m, i.e., about 16 m². Since the world is 10 × 10 m, the camera sees only about one-sixth of the area at any instant. In other words, perception remains local, and the system must move to build up the topological map.

As illustrated in Figure 4.2, we stream the full robot state from Gazebo to MATLAB/Simulink; the ground vehicle's position, velocity, and orientation; the aerial vehicle's linear position, linear velocity, angular position and angular velocity; gimbal mechanism's joint angles. We also send the camera images for the vision pipeline. In the return direction, MATLAB/Simulink sends the control outputs. The ground robot receives left- and right-wheel torque commands. The quadrotor receives four rotor torques. The gimbal receives three joint torques to keep the camera steady.

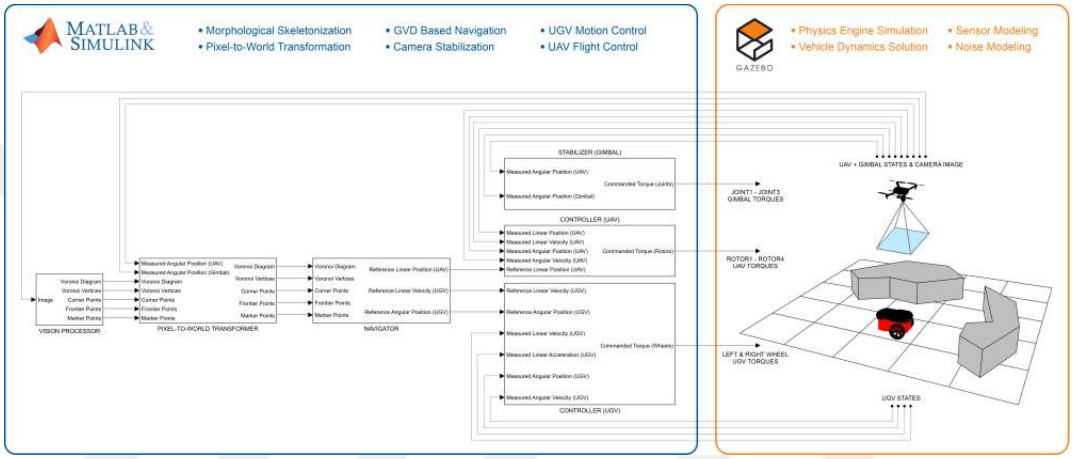
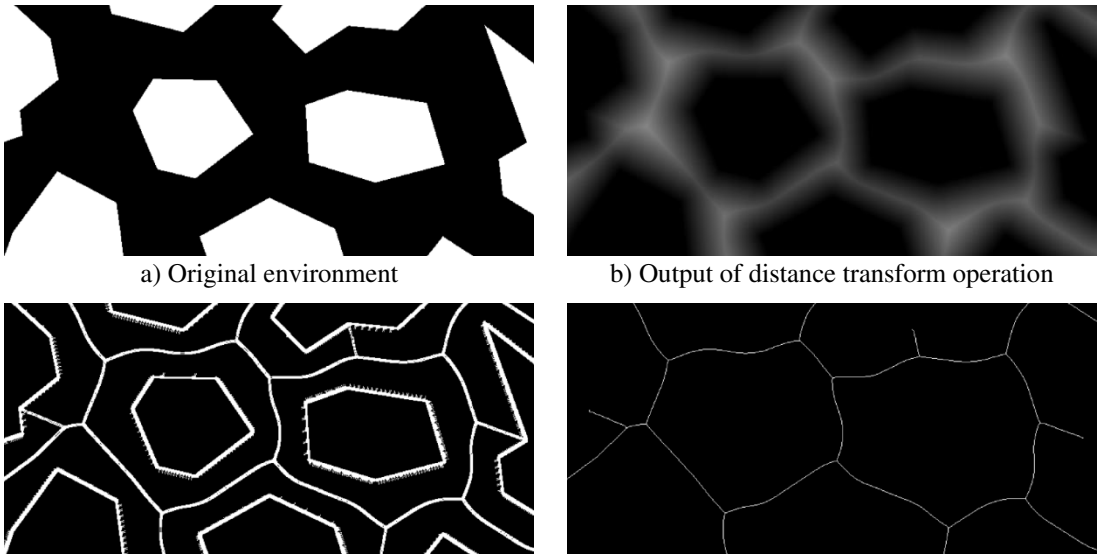


Figure 4.2 Co-simulation architecture for vision-based exploration integrating MATLAB/Simulink and Gazebo environments

4.3 Functional Blocks

This section reviews the main blocks in MATLAB/Simulink; Vision Processor, Pixel-to-World Transformer, Navigator, Stabilizer, and Controller as shown in Figure 4.2. We begin with the Vision Processor. The Vision Processor takes each camera frame and extracts the information needed for navigation and topology. It first marks free space in the image. It then computes the medial axis of that region, which we use as an approximate Generalized Voronoi Diagram (GVD). The diagram is an approximation because the pipeline works on pixels. We apply distance transforms, gradient filters, simple morphological operations, and finally skeletonization. This sequence is numerically stable and easy to run, but the result depends on image resolution. Figure 4.3 illustrates this pixel-based GVD approximation.



a) Original environment b) Output of distance transform operation

c) Output of gradient filtering d) Output of morphological filtering

Figure 4.3 Vision processing pipeline for Generalized Voronoi Diagram extraction

Distance computation process is the most important stage of this pipeline. For each pixel, we measure its distance to nearby obstacle boundaries; the Voronoi set then appears where those distances tie. This step supports the skeleton that we use downstream. From the skeleton, we extract several features. We keep the medial-axis pixels of the approximate GVD. We form a clustered set of *Voronoi Vertices* that mark junctions. We label *Corner Points* as endpoints that lie close to obstacles, and we label *Frontier Points* as endpoints far from obstacles, often near the image border. The Vision Processor also tracks the ground vehicle. It detects *Marker Points* fixed on the platform—one circular mark at the front and one rectangular mark at the rear—and uses their pixel locations to estimate pose (position and heading) in the image frame.

The second block, Pixel-to-World Transformer, turns all image features from pixel coordinates into the world frame. It uses only back-projection with known camera and gimbal geometry and the vehicle poses. In other words, we recover 3D points for 2D pixels under the stated spatial constraints. The inputs are the pixel sets for the Voronoi diagram, Voronoi vertices, corner points, frontier points, and markers points. Using vision system parameters, the block outputs their positions in world coordinates.

The Gazebo provides the aerial vehicle pose and the three gimbal joint angles at each step, so the camera pose in the world frame can be computed by a short chain of homogeneous transforms:

$$T_{wc} = T_{wq} \cdot T_{qg} \cdot T_{gc} = \begin{bmatrix} [R_{wc}]_{3x3} & [\vec{t}_w]_{3x1} \\ [0]_{1x3} & [1]_{1x1} \end{bmatrix} \tag{4.14}$$

where T_{wq} represents the world-to-quadcopter transformation that uses aerial vehicle position and orientation, T_{qg} denotes the quadcopter-to-gimbal transformation that uses kinematics equations and joint angles, and T_{gc} is the fixed gimbal-to-camera transformation. This composition yields the complete camera pose in world coordinates, where the R_{wc} rotation matrix provides camera orientation and the $\vec{t}_w$ translation vector provides camera position. For each pixel coordinate (u, v) in the image-plane, a corresponding direction vector in the camera-frame is constructed using the camera's intrinsic parameters:

$$\vec{d}_c = \begin{bmatrix} \frac{u - c_x}{f_x} & \frac{v - c_y}{f_y} & 1 \end{bmatrix}^T \tag{4.15}$$

where (c_x, c_y) represent the principal point coordinates and (f_x, f_y) are the focal lengths in pixel units. This direction vector is then transformed to world coordinates using the camera pose:

$$\vec{d}_w = R_{wc} \cdot \vec{d}_c \tag{4.16}$$

Since $\vec{d}_w$ is a unit direction vector in world coordinates, it must be translated and parameterized to form a complete ray equation in 3D space:

$$\vec{r}_w = \vec{t}_w + k \cdot \vec{d}_w \tag{4.17}$$

where $\vec{t}_w$ represents the camera position in world coordinates serving as the ray origin, $\vec{d}_w$ is the normalized direction vector that points from the camera center toward the corresponding feature in the scene, and k is the scalar parameter that determines the distance along the ray. To find the intersection point of this ray with

any object, the parameter k must be computed by solving the ray-plane intersection equation:

$$k = \frac{z_{target} - t_{w,z}}{d_{w,z}} \tag{4.18}$$

where z_{target} is the target plane height, $t_{w,z}$ and $d_{w,z}$ are the z-components of the camera position and direction vector, respectively. The process is applied to each image-space feature (e.g., Voronoi Vertices, Corner Points, Frontier Points, Marker Points) to compute its corresponding world-frame coordinates, assuming it lies on either the ground plane (z_{target} = 0) or a known constant-height plane (z_{target} = 0.27 [m] vehicle height). This homography-based approach provides computationally efficient transformation while maintaining sufficient accuracy for navigation tasks.

The third component, Navigator block, implements the vision-based GVD exploration policy and coordinates the UAV-UGV team to cover the environment. It reads world-frame features from the Pixel-to-World Transformer (the Voronoi diagram, Voronoi vertices, corner points, frontier points, and marker points), together with the estimated UGV pose. From these inputs it issues reference commands: linear velocity and angular position for the ground vehicle, and target positions for the aerial vehicle. Target selection follows a simple priority rule as listed in Algorithm 2. If the camera view contains unvisited corner points, the Navigator chooses one of them first; corners usually sit on obstacle boundaries and often lead to shorter moves. If no corners are visible, it prefers unvisited Voronoi vertices, since vertices are junctions that open multiple routes. When neither corners nor vertices are in view, it heads toward frontier points near the image boundary to reach unseen space. As the ground vehicle moves, new corners or vertices appear; the same priority is applied again, so exploration continues in an orderly way. The Navigator also keeps a dynamic vertex list that records visited junctions and per-edge completion flags.

Algorithm 2 Aerial Camera-based GVD Exploration Algorithm

```
Initialize empty VertexList, Set UAV reference altitude, Set gimbal stabilization
Find closest point on Voronoi Diagram to UGV position
Generate linear velocity and heading references for UGV
Estimate UGV position using marker point tracking
Set estimated UGV position as reference for UAV
while untraced directions exist in VertexList do
    Detect available points in current camera field of view
    if unvisited Corner Points exist in field of view then
        target ← closest unvisited Corner Point
        if distance to corner point ≤ threshold then
            Mark corner point as visited
            Initiate reverse movement to return to vertex
        end if
    else if unvisited Voronoi Vertices exist in field of view then
        target ← closest unvisited Voronoi Vertex
        if UGV reaches target vertex then
            if target ∉ VertexList then
                Add target to VertexList with available directions (angles)
            end if
            Mark approach direction as traced (flag = 1)
        end if
    else if Frontier Points exist in field of view then
        target ← closest Frontier Point
        Monitor for frontier point transformation:
        if frontier point disappears AND vertex appears at same location then
            Apply Voronoi Vertex algorithm steps
        else if frontier point disappears AND corner point appears then
            Apply Corner Point algorithm steps
        end if
    end if
    Generate linear velocity and heading references for UGV
    Estimate UGV position using marker point tracking
    Set estimated UGV position as reference for UAV
end while
return Exploration complete - all accessible regions covered
```

Figure 4.4 sketches how exploration unfolds. In panel (a), the robot has already visited v_1 through v_8 and the corner points c_1 and c_2. It has also traced paths toward unseen areas using frontier cues f_1- f_5. These completed vertices and paths appear as yellow lines and outline the portion of the topology covered so far. At v_8, the planner applies its priority rule to pick a new target. It checks for corner points connected to v_8 first; none are available. It then looks for Voronoi vertices reachable from v_8. Two candidates show up: the already visited v_7 below and an unnumbered vertex to the left. Because unvisited items take precedence, the system chooses the

left option and issues the corresponding linear velocity and angular position references to steer the ground vehicle in that direction.

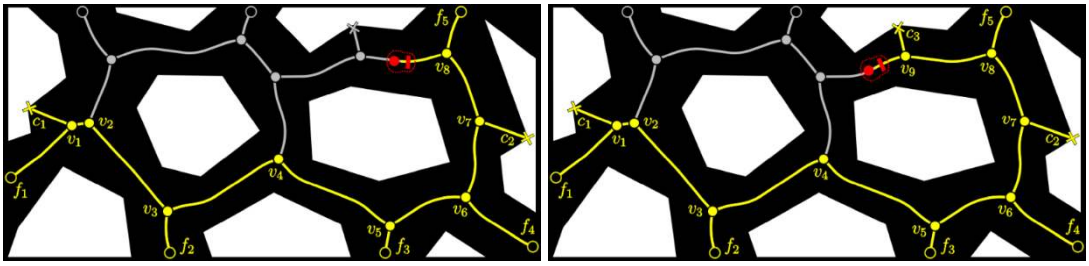


Figure 4.4 Sequential progression of aerial camera-based exploration algorithm

A key difference of the vision-based case appears in the visuals. In Figure 3.3, untraced paths are drawn as dashed lines, and unvisited vertices never appear because they are still unknown. In Figure 4.4 (a), by contrast, untraced paths are shown as solid lines and unvisited vertices show up as unnumbered points. The camera already sees these features. The planner therefore knows they exist even before the robot reaches them. Exploration then becomes a simple routine: visit what is visible, give it a label, and record it. In Figure 4.4 (b), the ground vehicle reaches an unnumbered Voronoi vertex and names it v_9. While at v_9, the priority rule first points to the nearby corner; that point is labeled c_3. The robot returns to v_9 afterward. With no unvisited corners left at this vertex, the rule picks an unvisited Voronoi vertex—the one to the left—and the system issues linear velocity and angular position references to move that way. If neither corners nor vertices were available at the current pose, the planner would target a frontier that marks an untraced path. This process repeats until every path has been followed, yielding complete coverage.

The fourth component, the UAV Controller block, keeps the aerial vehicle on target. It takes position setpoints from the Navigator and state feedback from Gazebo: linear position and velocity, and attitude states (roll, pitch, yaw) with their angular rates. From these signals it computes four rotor torques and sends them back to Gazebo for actuation. The control architecture employs a cascaded multi-loop structure that decomposes the complex 3D positioning task into manageable control subproblems while addressing the inherent coupling between translational and rotational dynamics

in quadrotor systems through hierarchical control implementation. This architecture can be analyzed through three primary subsystems: the XY-Position Controller, the Attitude Controller, and the Altitude Controller, as illustrated in Figure 4.5.

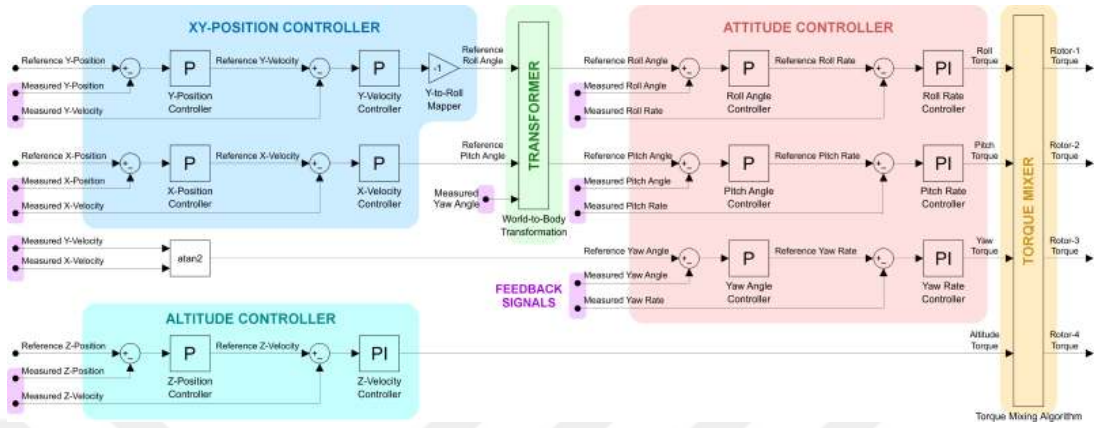


Figure 4.5 Controller (UAV) block architecture showing cascaded multi-loop control structure with separate subsystems for position and attitude control

The XY-Position Controller employs a cascaded P+P (proportional + proportional) structure, while the Attitude Controller and Altitude Controller utilize P+PI (proportional + proportional-integral) configurations, similar to the ground vehicle control methodology presented earlier. The XY-Position Controller generates attitude reference commands in the world coordinate frame, which require transformation to the body coordinate frame for attitude control implementation through the World-to-Body Transformation block, as described by the following equations:

$$\begin{bmatrix} \phi_{ref}^{B} \\ \theta_{ref}^{B} \end{bmatrix} = \begin{bmatrix} \cos(\psi_{mea}) & \sin(\psi_{mea}) \\ -\sin(\psi_{mea}) & \cos(\psi_{mea}) \end{bmatrix} . \begin{bmatrix} \phi_{ref}^{W} \\ \theta_{ref}^{W} \end{bmatrix} \tag{4.19}$$

where ϕ_{ref}^{B} and θ_{ref}^{B} represent the roll and pitch reference angles in the body coordinate frame, ϕ_{ref}^{W} and θ_{ref}^{W} denote the roll and pitch reference angles in the world coordinate frame, and ψ_{mea} is the measured yaw angle of the UAV. The rotation matrix accounts for the vehicle's current heading so attitude commands line up across frames. In parallel, the Altitude Controller runs its own loop to hold the target height. The torques from the position and attitude loops then pass through a torque-

mixing stage, which spreads the control effort across the four rotors to produce the commanded motion. The allocation is given by the following equations:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \tau_{roll} \\ \tau_{pitch} \\ \tau_{yaw} \\ \tau_{alt} \end{bmatrix} \tag{4.20}$$

where τ_1, τ_2, τ_3, and τ_4 represent the individual rotor torque commands transmitted to rotors 1-4 respectively, while τ_{roll}, τ_{pitch}, τ_{yaw}, and τ_{alt} denote the control torques generated by the respective controller subsystems. The mixing matrix consists of +1 and -1 elements that determine whether each controller's contribution should be added to (accelerating the rotor) or subtracted from (decelerating the rotor) the total torque command for each individual rotor. The matrix configuration corresponds to the X-configuration quadrotor layout with Rotor-1 positioned at front-right, Rotor-2 at rear-left, Rotor-3 at front-left, and Rotor-4 at rear-right, operating within the NWU (North-West-Up) coordinate system employed in the Gazebo simulation environment. This systematic torque distribution ensures proper coordination between individual rotor commands and the desired vehicle motion in three-dimensional space.

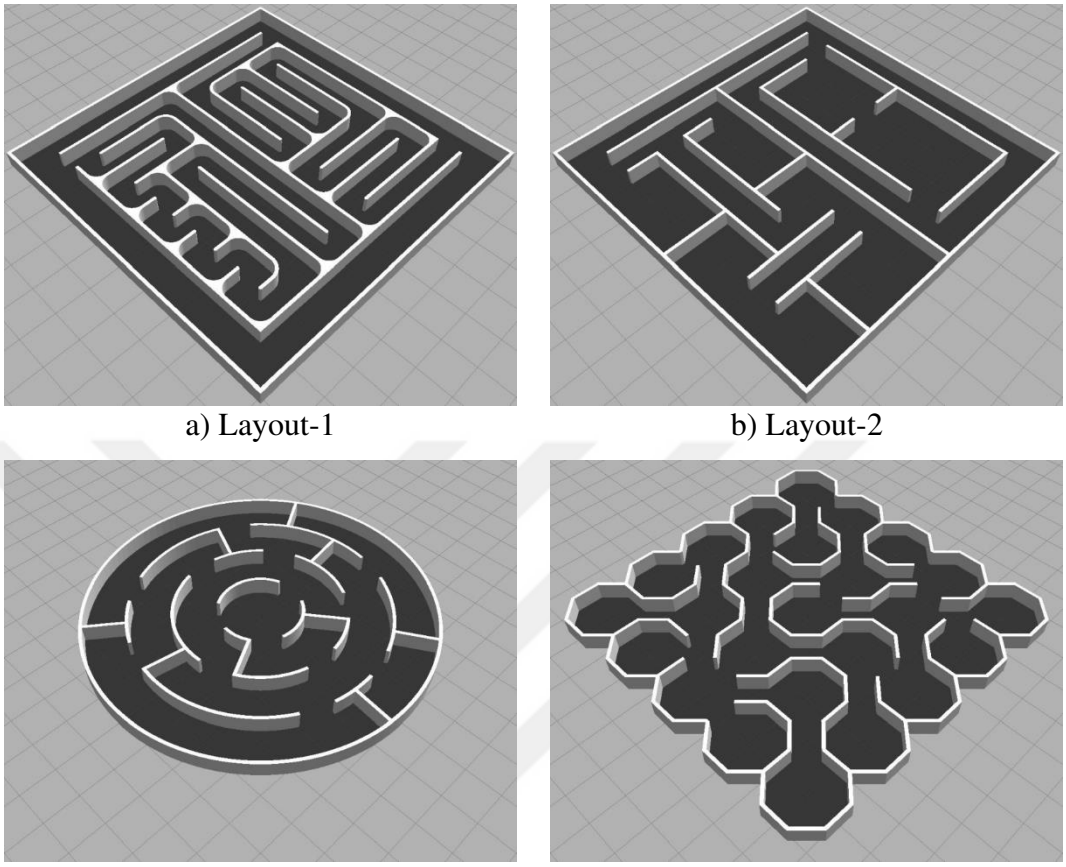
The Controller (UGV) block uses the same cascaded design and equations as in Section 3.3. It takes the Navigator's linear velocity and angular position references and outputs left- and right-wheel torques.

The Stabilizer (Gimbal) block keeps the camera pointed straight down, regardless of the UAV's attitude. This consistent alignment is essential for reliable vision sensing. The block reads the UAV's roll, pitch, and yaw, together with the three gimbal joint angles from Gazebo, and returns the torque commands for each joint to hold a 90° downward view. Stabilization has two parts. First, we accomplish gravity compensation: using the center-of-mass of each gimbal link and the camera, the block calculates the static torques needed to balance the payload at the current orientation. Then a set of PD controllers, one per joint, corrects dynamic errors during motion. Together, these steps counter the UAV's rotations and keep the camera stable throughout the mission.

CHAPTER 5
SIMULATION RESULTS

As detailed in the preceding sections, a simulation framework is developed to enable topological map construction in unknown environments, employing two distinct methods: Section 3 presents a range sensor-equipped ground vehicle, while Section 4 introduces a cooperative architecture featuring an aerial vehicle that provided visual navigation support to the ground vehicle via a downward-facing camera. This section reports a comparative evaluation of the simulation results obtained from both approaches. The primary objective is to identify the measurable differences arising from the distinct sensing modalities (range sensor vs. aerial camera) and the algorithmic behavior of each method under different environmental conditions. In particular, the analysis focuses on the systems' ability to achieve complete topological exploration in the presence of increasing spatial complexity. For this purpose, four distinct test environments are constructed within the Gazebo simulation platform and are illustrated in Figure 5.1.

The four layouts exhibit distinct spatial configurations: Layout-1 consists of narrow corridors arranged in a looped track-like formation; Layout-2 includes multiple rectangular walls placed in a loosely structured grid pattern; Layout-3 features circular obstacles organized in concentric paths; and Layout-4 comprises interconnected octagonal cells forming a modular tiling layout. To enhance visual clarity for the aerial camera system, obstacles are rendered in a light (white) color against a dark (gray) background, ensuring high-contrast imagery during the image processing phase. All environments share a consistent geometric specification: obstacle walls are 0.5 meters in height and 0.1 meters in thickness.



a) Layout-1 b) Layout-2

c) Layout-3 d) Layout-4

Figure 5.1 Visual representation of the four test environments constructed in the Gazebo simulation platform

We build a ground-truth GVD for each test environment to compare the two methods fairly. We begin with MATLAB's *voronoi(x,y)* routine. In its default form the function expects point sites, which matches the classical diagram in Section 2.1. Our obstacles have area, so we approximate the generalized case in Section 2.2 by placing a dense set of samples along every obstacle boundary from its known geometry and then running *voronoi(x,y)* on those samples. The raw output includes edges that cross into obstacles or lie completely inside them because the point-site model ignores object size. We remove these in a simple post-process: check each Voronoi edge against the obstacle boundaries and discard the invalid ones. This yields a valid generalized Voronoi skeleton for each environment. From the cleaned

diagram we also record the number of Voronoi vertices and corner points for use in the analysis.

For evaluation we track three aspects: spatial accuracy (vertex-location error and path-tracking error), path efficiency (redundancy ratio and total distance), and temporal efficiency (total exploration time).

All simulations employed a fixed sampling time of 0.005 seconds, providing uniform temporal resolution for data acquisition and control updates. Additionally, while the Pioneer 2-DX platform supports translational speeds up to 2.2 m/s and rotational speeds up to 360 deg/s according to manufacturer specifications, the maximum speeds are limited to 0.8 m/s and 180 deg/s respectively in the simulation scenarios to maintain stable navigation control and ensure reliable sensor data processing.

The comparative performance analysis is illustrated through a systematic visual presentation in Figures 5.2, 5.3, 5.4, and 5.5, which demonstrate the exploration results for each layout configuration.

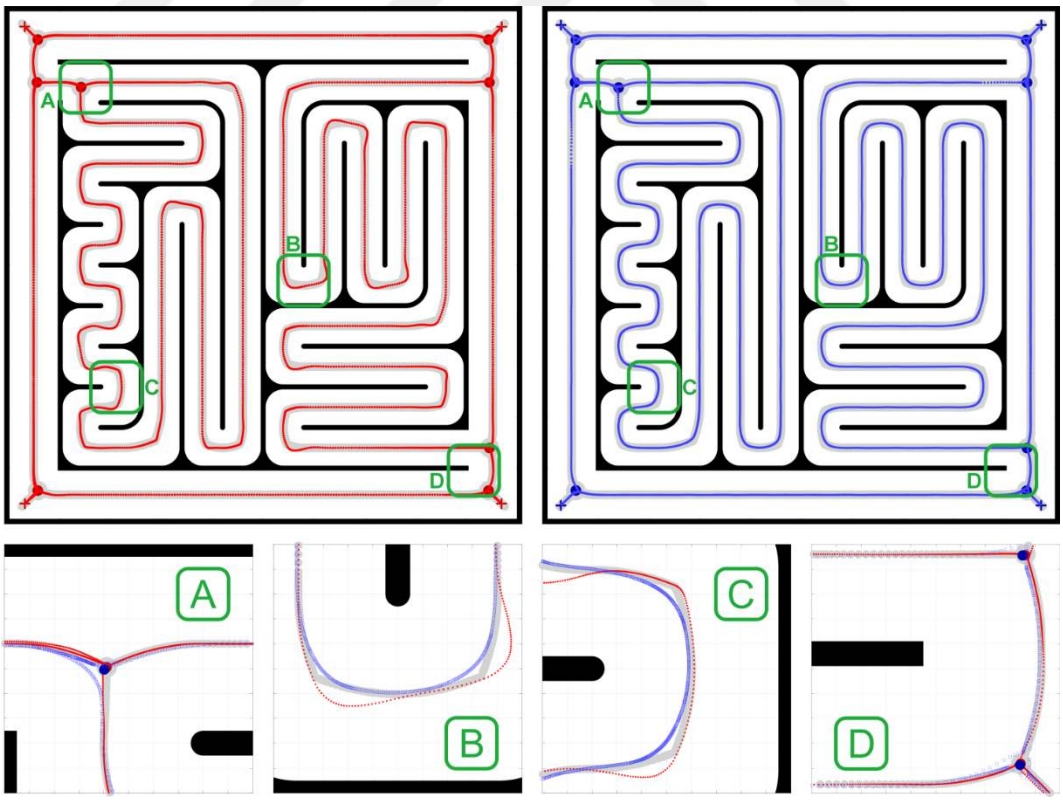


Figure 5.2 Exploration results for Layout-1 (track-like configuration with 8 vertices and 4 corner points)

All figures use a fixed three-panel layout to make side-by-side comparisons easy. Top-Left Panel: the ground-truth GVD is drawn in gray and the range-sensor run is overlaid. The robot's path appears as red dots. Detected Voronoi vertices are dark-red filled circles, and corner points are dark-red crosses. Top-Right Panel: the same ground-truth layer stays in gray, now compared with the aerial-camera run. The robot path is shown as blue dots, with Voronoi vertices as dark-blue filled circles and corner points as dark-blue crosses. Bottom Panel: zoomed insets (frames A–D) show selected regions. Here we overlay all three data sets; ground truth, range-sensor, and aerial-camera. In these close-ups, the aerial-camera samples are drawn as blue squares for extra separation. This standardized visualization approach enables systematic evaluation of spatial accuracy, feature detection completeness, and trajectory characteristics across all experimental configurations, providing a comprehensive foundation for quantitative performance assessment.

The track-like configuration of Layout-1 featuring 8 Voronoi vertices and 4 corner points as given in Figure 5.2, represents a topologically straightforward environment that enables clear evaluation of fundamental algorithmic differences. Although both approaches successfully achieve complete feature detection, significant differences emerge in spatial accuracy performance. The range sensor-based approach demonstrates superior vertex localization precision with a mean error of 2.1×10^{-3} meters compared to 6.2×10^{-3} meters for the aerial camera-based method. This three-fold accuracy advantage stems from two key factors: the direct utilization of range measurements without coordinate transformations, and the geometric computation of vertex positions as circumcenters of equidistant obstacle points, enabling sub-resolution accuracy beyond the sensor's nominal 0.01-meter precision. Conversely, the aerial camera-based approach faces inherent limitations from its 640x480 pixel resolution, which constrains image processing results to pixel-level precision. Furthermore, the conversion of 2D image plane coordinates to 3D world coordinates through back-projection introduces additional uncertainty, compounded by potential gimbal stabilization errors despite kinematic compensation during aerial vehicle maneuvers. Path tracking accuracy follows similar patterns, with the range sensor approach achieving lower mean tracking error (15.2×10^{-3} vs 19.0×10^{-3} meters) for

the same technical reasons governing vertex localization precision. However, an unexpected reversal occurs in maximum path tracking error, where the aerial camera-based approach exhibits better performance (57.0×10^{-3} meter) compared to the range sensor approach (91.0×10^{-3} meter). This advantage reflects the predictive nature of vision-based navigation: upcoming turns and features are detected in advance, enabling preemptive deceleration. In contrast, the range sensor-based approach operates reactively, detecting directional changes only after they commence, resulting in higher-speed cornering and greater maximum deviations. This phenomenon is clearly illustrated in Region-B and Region-C of Figure 5.2, where the greater spacing between red points compared to blue squares indicates higher vehicle velocities during turns, as both methods employ identical and fixed sampling frequencies.

Path efficiency analysis reveals substantial advantages for the aerial camera-based approach, with a redundancy ratio of 1.39 compared to 1.68 for the range sensor-based method. This 17% efficiency improvement results from strategic corner-priority exploration planning. When reaching a Voronoi vertex, the aerial camera system visualizes connected vertices and corner points, enabling systematic corner point completion before proceeding to distant vertices. The range sensor approach, however, employs angle-based path selection without knowledge of destination types, frequently leaving corner points unvisited and necessitating extensive backtracking after completing distant explorations. The most pronounced performance differences appear in exploration time, with the aerial camera approach requiring 1240.5 seconds (≈ 20.7 minutes) compared to 2619.7 seconds (≈ 43.7 minutes) for the range sensor—representing a 2-fold reduction. This advantage stems from two contributing factors. First, exploration time correlates directly with path redundancy—the aerial camera's 17% reduction in total distance (redundancy ratio 1.39 vs 1.68) correspondingly reduces temporal requirements. Second, more efficient velocity profiles enhance performance through predictive navigation. The aerial camera's advance knowledge of target locations enables precise deceleration timing and efficient stopping control. Conversely, the range sensor approach must operate conservatively, uncertain of exact vertex locations or the number of equidistant objects, resulting in premature deceleration and extended low-speed operation. This conservative behavior is evident in Region-A of Figure 5.2, where densely clustered

red points near the vertex indicate prolonged low-speed maneuvering compared to the more sparse blue square distribution.

These performance metrics for Layout-1, along with comparative results for all other layouts, are presented in Table 5.1 and Table 5.2.

Table 5.1 Performance Analysis: Spatial Accuracy Comparison

Layout	Method	Number of VV and CP	Vertex Location Mean Error [m]	Path Tracking Mean Error [m]	Path Tracking Max Error [m]
1	RS	8/8, 4/4	**0.0021**	**0.0152**	0.0910
	AC	8/8, 4/4	0.0062	0.0190	**0.0570**
2	RS	**28/28**, 30/30	**0.0015**	**0.0115**	0.0855
	AC	26/28, 30/30	0.0083	0.0152	**0.0527**
3	RS	39/39, 31/31	**0.0018**	**0.0118**	0.0824
	AC	39/39, 31/31	0.0073	0.0157	**0.0547**
4	RS	**39/39**, 110/110	**0.0020**	**0.0133**	0.0726
	AC	28/39, 110/110	0.0097	0.0174	**0.0583**

***RS:** Range-Sensor, **AC:** Aerial Camera, **VV:** Voronoi Vertex, **CP:** Corner Points*

Table 5.2 Performance Analysis: Path Efficiency and Exploration Time Comparison

Layout	Method	Actual Path Length [m]	Explored Path Length [m]	Redundancy Ratio	Time [s]
1	RS	113.84	190.73	1.68	2619.7
	AC		**158.24**	**1.39**	**1240.5**
2	RS	99.76	182.56	1.83	5130.8
	AC		**147.65**	**1.48**	**1651.3**
3	RS	79.94	157.48	1.97	5832.4
	AC		**127.90**	**1.60**	**1859.8**
4	RS	147.35	324.17	2.20	7132.3
	AC		**254.92**	**1.73**	**2161.5**

***RS:** Range-Sensor, **AC:** Aerial Camera*

The remaining layouts demonstrate progressively increasing topological complexity, providing a comprehensive evaluation framework for algorithmic scalability assessment. Layout-2 contains 28 Voronoi vertices and 30 corner points, Layout-3 features 39 Voronoi vertices and 31 corner points, while Layout-4 presents the highest complexity with 39 Voronoi vertices and 110 corner points. Examination of Table 5.1 and Table 5.2 results across all layouts reveals consistent performance patterns that validate the fundamental characteristics observed in Layout-1. The range sensor-based approach maintains superior spatial accuracy throughout all configurations, consistently achieving lower Voronoi vertex location errors and path tracking mean errors. Similarly, the aerial camera-based approach demonstrates systematic advantages in path tracking maximum error across all layouts, reflecting its predictive navigation capabilities. Path efficiency and temporal efficiency analysis

confirm the aerial camera-based approach's superiority across all environmental configurations. The performance gap becomes increasingly pronounced with topological complexity: redundancy ratios improve from 17% in Layout-1 to 21% in Layout-4, while temporal efficiency advantages escalate from 2-fold in Layout-1 to 3-fold in Layout-4. This scaling behavior demonstrates that the strategic planning advantages of aerial camera-based exploration become more significant as environmental complexity increases.

However, a critical limitation emerges for the aerial camera-based approach regarding vertex detection completeness in certain layout configurations. In Layout-2 and Layout-4, this approach misses 2 and 11 vertices respectively due to the close proximity of multiple vertices that exceed the method's spatial resolution capabilities. This detection deficit stems from the same technical limitations responsible for higher vertex location errors discussed previously. When multiple ground truth Voronoi vertices exist in close proximity as illustrated in Region-A and Region-B of Figure 5.3. The combination of pixel-level resolution constraints and coordinate transformation uncertainties causes these distinct vertices to be perceived as a single point.

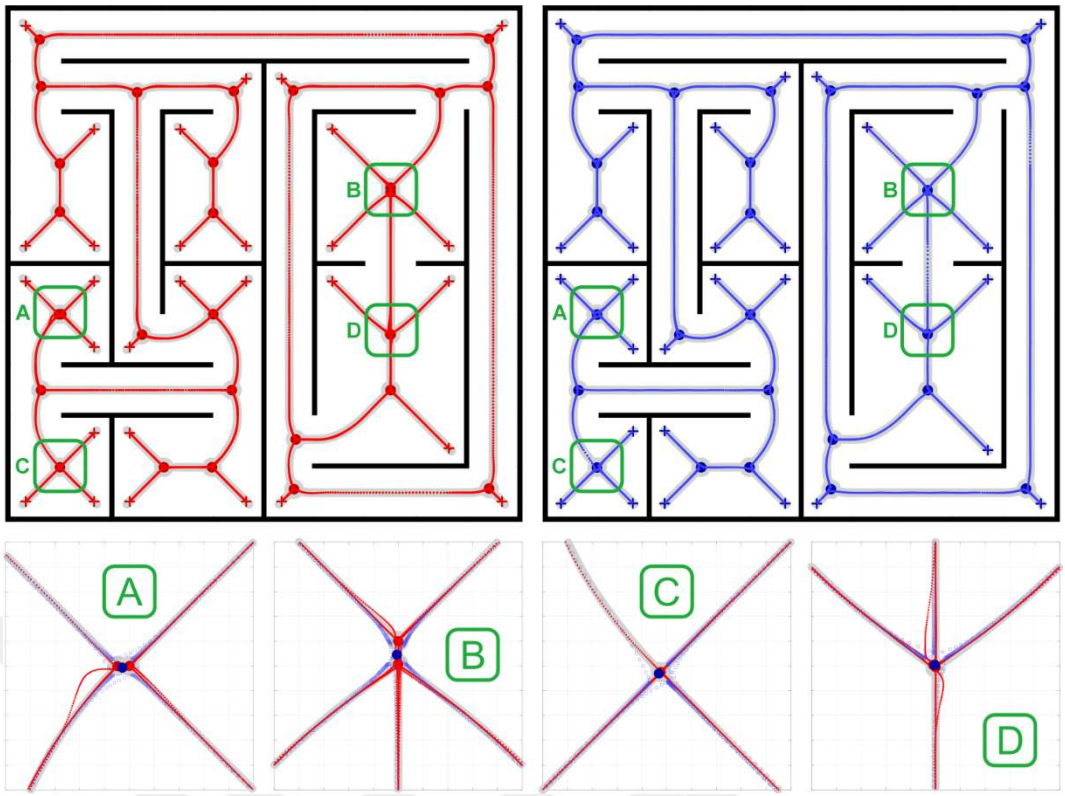


Figure 5.3 Exploration results for Layout-2 (grid pattern with rectangular walls containing 28 vertices and 30 corner points)

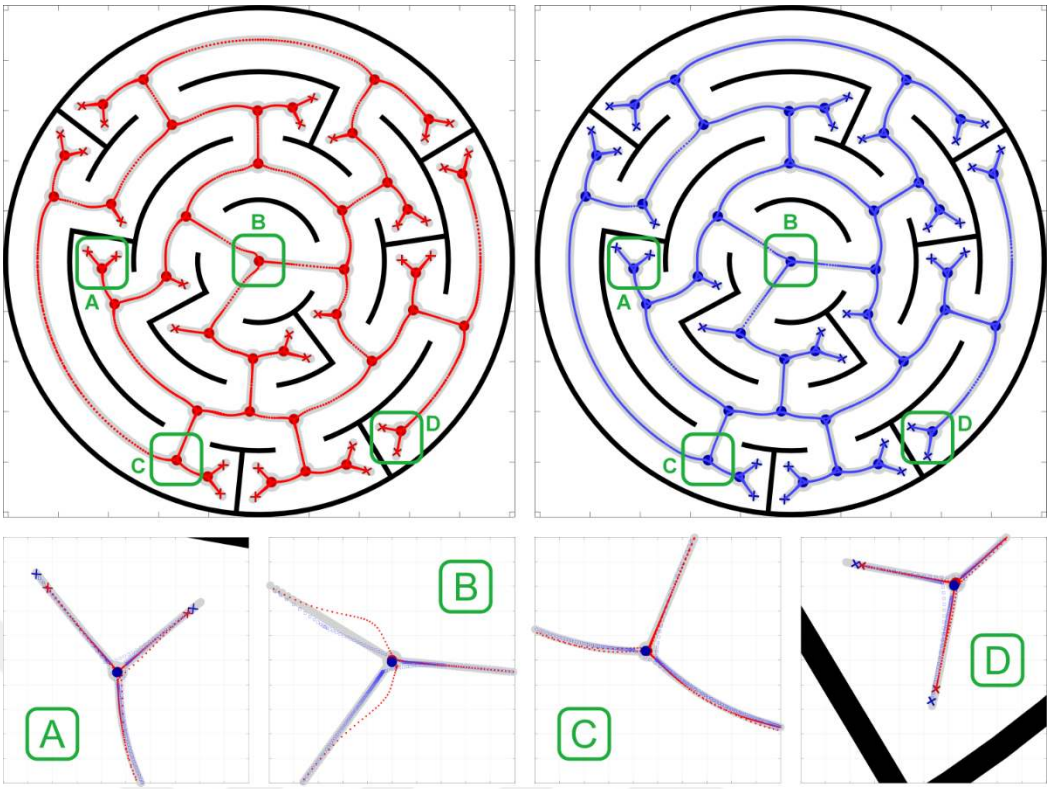


Figure 5.4 Exploration results for Layout-3 (circular obstacles in concentric arrangement with 39 vertices and 31 corner points)

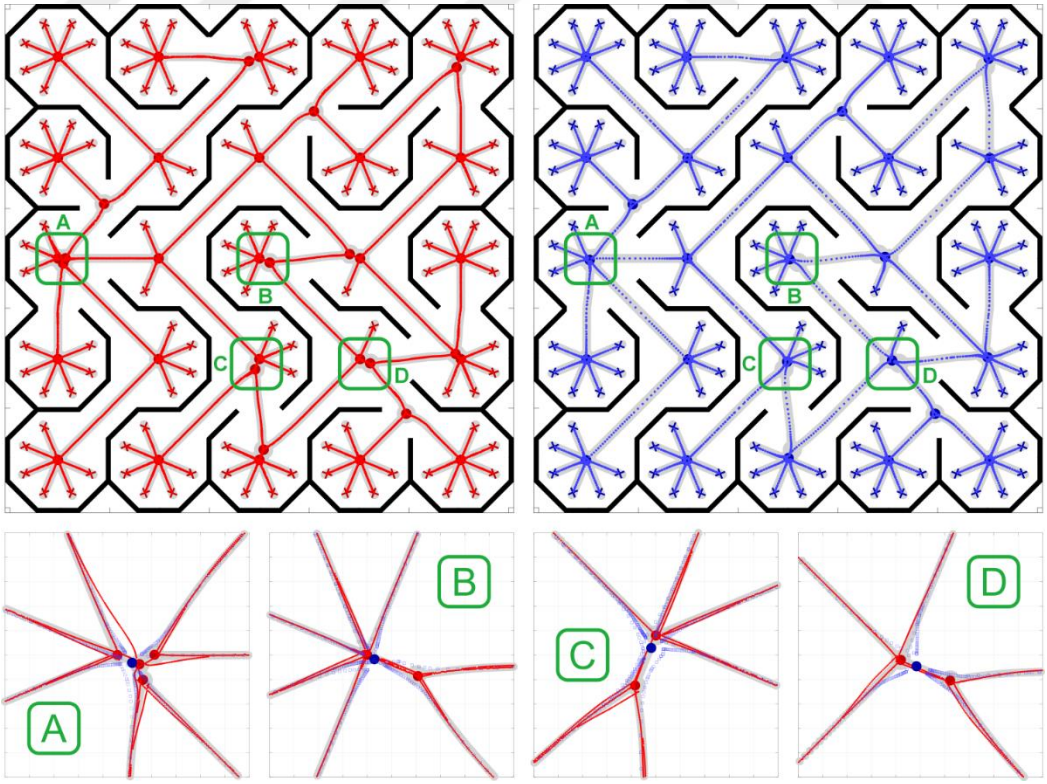


Figure 5.5 Exploration results for Layout-4 (octagonal cell configuration with 39 vertices and 110 corner points)

This phenomenon becomes more pronounced in Layout-4, where Figure 5.5 demonstrates extensive vertex clustering across all detailed regions (A, B, C, and D), with multiple closely spaced vertices consistently interpreted as singular features. Crucially, the incomplete vertex detection does not compromise topological map completeness or exploration coverage. All paths are successfully traversed, and complete environmental exploration is achieved in every layout. The detection limitation affects only the granularity of topological representation—paths that should theoretically be defined by multiple distinct vertices are instead represented through unified single-vertex definitions, while maintaining full connectivity and coverage integrity.

To provide visual evidence of the exploration efficiency differences, Figure 5.6 presents simulation screenshots from Layout-2 captured at sequential time instances throughout the exploration process. The figure demonstrates the temporal progression of both methodologies, clearly illustrating the performance differences between predictive and reactive navigation approaches. Subfigures 5.6 (a)-(d) show the aerial camera-based exploration systematically advancing through the environment, achieving complete coverage at t_3 = 1651 s. The blue trajectory lines reveal movement patterns of strategic corner-priority target selection, where advance knowledge of topological features allows systematic completion of corner points before proceeding to distant vertices, minimizing backtracking requirements. Subfigures 5.6 (e)-(l) present the range sensor-based exploration requiring substantially longer completion time of t_7 = 5131 s. The red trajectory demonstrates extensive backtracking due to angle-based path selection without knowledge of destination types. It is clear that upon completion of the aerial camera-based exploration at t_3 = 1651 s, the range sensor-based approach has not yet explored even half of the environment. This visualization directly supports the quantitative findings about time efficiency presented in Table 5.2, where the combination of strategic planning and optimized velocity control enabled the aerial camera approach to achieve 3-fold advantage across all test configurations.

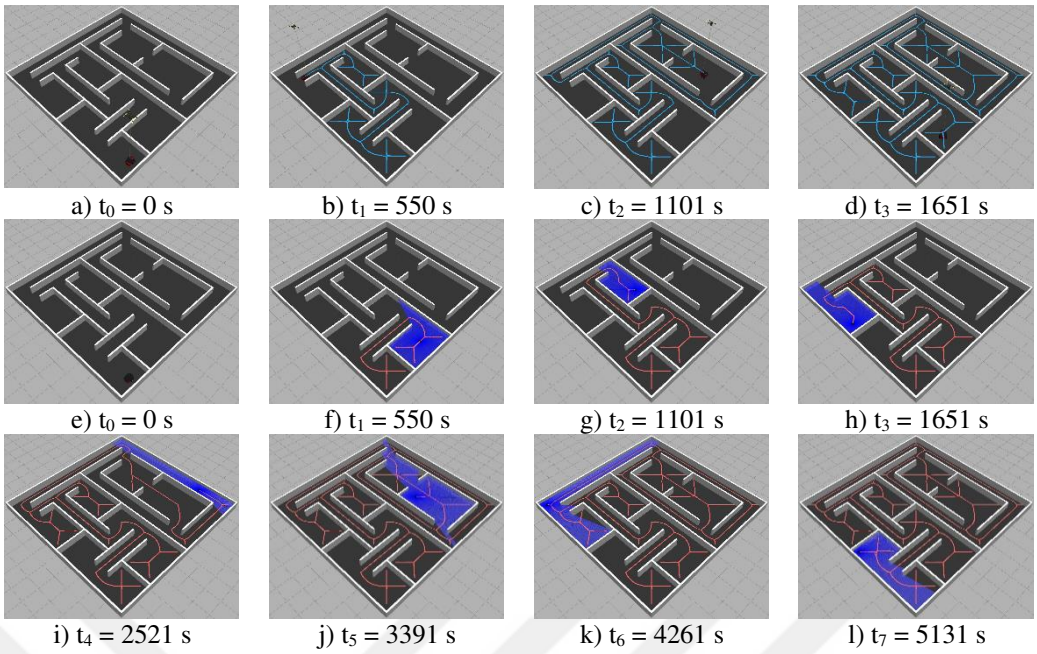


Figure 5.6 Simulation screenshots of comparative progression in Layout-2 illustrating time efficiency differences between methodologies

To assess overall performance under different operational priorities, composite metrics are calculated using weighted scoring approaches. Under accuracy-priority weighting (80% spatial precision), the range sensor approach achieved superior composite scores across all layouts, reinforcing its fundamental precision advantages in vertex localization and path tracking. Democratic weighting (equal importance to all metrics) similarly favored the range sensor methodology due to the combined influence of its accuracy benefits. Conversely, time-priority weighting (55% temporal performance) consistently demonstrated the superiority of the aerial camera approach across all environmental configurations, highlighting the significant impact of its exploration efficiency advantages when temporal performance is prioritized.

CHAPTER 6
DISCUSSION

The following discussion critically examines the fundamental trade-offs between the range sensor-based and aerial camera-based exploration approaches, analyzing their respective advantages and limitations while considering practical implementation aspects that complement the quantitative performance metrics. The range-sensor method faces several algorithmic challenges that drive how it operates. First comes boundary finding: from raw range scans it must infer object outlines and then pick the nearest contact on each object by detecting local minima. In practice, mixed geometry (rectangles, circles, convex and non-convex shapes) makes the task harder. The algorithm must tell whether close measurements come from different objects or from two faces of the same non-convex object. The problem grows when smooth curves are represented by short line segments, which create extra local minima. A more critical challenge emerges in the detection of Voronoi vertices, which are theoretically defined as points equidistant to three or more obstacles. With perfect data you would look for exact equalities. Real sensors add noise, so exact ties never occur. The implementation therefore uses a tolerance: a point counts as a vertex when several ranges fall within a small band of one another. This creates uncertainty during approach. The ground vehicle cannot know in advance how many obstacles will meet the tolerance at the final position. As it moves, some ranges drift into the band while others drift out, and the “vertex condition” changes on-the-fly. To avoid overshoot, the policy slows down strongly near candidate vertices. That caution increases total exploration time which is consistent with our quantitative results. However, this operational constraint paradoxically contributes to the method's superior spatial accuracy. The combination of reduced vehicle velocity with geometric computation of Voronoi vertices enables exceptionally accurate localization across all tested configurations. Once a set of near-equal ranges is available, the algorithm computes the vertex as the center of a circle through three equidistant obstacle points (or a best-fit circle if more than three points qualify).

With the robot already moving slowly, this calculation yields very tight estimates—often below the LiDAR's nominal 0.01 m resolution.

On the other hand, the camera-based method inherits limits from vision and from the image pipeline. A finite pixel grid caps spatial precision: the GVD and its feature (Voronoi vertices and corner points) can be located only to about a pixel. That cap is not the only source of error. We must also lift 2D pixels into 3D by back-projection. Doing so brings in uncertainty from the UAV's pose, the gimbal kinematics, and the joint angle readings. Each link in that chain adds a little noise; together they reduce accuracy compared with direct range measurements. In practice we see larger vertex-location errors—often several times those of the range sensor—and, in tight layouts, missed vertices when multiple ground-truth vertices lie closer than the method can resolve. The result is a coarser topological graph: some nearby vertices collapse into one. Even so, connectivity and overall coverage remain intact; every traversable path is still explored. These costs come with real benefits. Vision lets the system see ahead. The planner can spot upcoming vertices before the robot reaches them, shape the speed profile, and time deceleration precisely. Unlike the range-sensor policy—forced to creep near uncertain vertices—the vision system can cruise at higher speed and slow only near known targets, avoiding long low-speed phases. The camera's field of view also reveals several outgoing connections at each junction. That context enables a simple priority rule—corners first, then unvisited vertices, then frontiers—that prevents backtracking and duplicate travel. Across all layouts, this look-ahead property and the corner-priority rule drive lower redundancy ratios and shorter mission times, even though per-vertex accuracy is lower than with direct ranging.

The implementation of both exploration methodologies necessitates the specification of multiple design parameters, though their nature and environmental dependencies differ substantially. For the range-sensor approach, several settings directly shape behavior. Finding the nearest contact on each object needs more than a simple local-minimum scan, especially with non-convex shapes or circles; extra filtering is added to reject false positives, and those filters must be tuned according to the scene's geometry. Because range noise makes exact ties impossible, the algorithm uses an equality threshold to decide when multiple measurements count as "the same distance." That threshold is calibrated from sensor precision and typical scene

variability. A second tolerance is needed when the robot revisits a vertex: new measurements rarely match the stored coordinates exactly, so a spatial tolerance decides whether two detections refer to the same vertex. Set it too tight and you split one vertex into duplicates; too loose and you merge distinct ones. Both tolerances depend on the sensor's accuracy and on how dense and complex the local topology is. The aerial-camera pipeline brings a different set of difficulties. Gradient and morphological filters drive the pixel-based GVD, and their thresholds depend on what the image looks like in practice: the contrast between floor and obstacles, lighting changes, and shadows. These parameters are adjusted so that edges and free space remain stable across frames, keeping feature extraction reliable even as the scene appearance varies.

CHAPTER 7
CONCLUSION

This study has presented a comparative analysis of two distinct methodologies for autonomous exploration and topological map extraction using Generalized Voronoi Diagrams in unknown environments. Through systematic evaluation across four increasingly complex layouts, the investigation has revealed fundamental trade-offs between spatial accuracy and operational efficiency that characterize range sensor-based and aerial camera-based exploration paradigms. The simulation results demonstrate that the range sensor-based approach achieves superior spatial precision, with vertex localization errors of 1.5×10^{-3} - 2.1×10^{-3} meters (compared to 6.2×10^{-3} - 9.7×10^{-3} meters for the aerial camera approach)—approximately three to five times lower than the vision-based methodology. This accuracy advantage reflects both sub-resolution vertex localization through geometric computation and the absence of coordinate transformation errors. However, this precision comes at substantial operational cost, with exploration times of 2619.7-7132.3 seconds (versus 1240.5-2161.5 seconds for aerial camera) and redundancy ratios reaching 1.68-2.20 (compared to 1.39-1.73). The reactive nature of range-based navigation, characterized by conservative velocity profiles and uncertainty in vertex detection, necessitates extended periods of low-speed operation that significantly impact temporal efficiency.

Conversely, the aerial camera-based approach demonstrates remarkable operational advantages through predictive navigation capabilities enabled by its broader spatial context. Despite spatial accuracy limitations inherent to pixel-level resolution (640×480) and back-projection uncertainties—manifesting in vertex location errors three to five times higher than range sensing and incomplete vertex detection when multiple vertices exist in close proximity—the method achieves superior operational efficiency. Notably, while 2 and 11 vertices remain unresolved in Layout-2 and Layout-4 respectively due to insufficient spatial resolution to distinguish closely spaced topological features, this limitation does not compromise the completeness of

topological mapping as all paths are successfully traversed and full environmental coverage is achieved. The redundancy ratios of 1.39-1.73 (versus 1.68-2.20 for range sensor) and exploration time reductions by factors of 2-3 demonstrate the effectiveness of strategic planning capabilities afforded by simultaneous visibility of multiple topological features.

The comparative analysis further reveals distinct dependencies on design parameters and environmental conditions. The range sensor approach requires empirical tuning of multiple threshold parameters for vertex detection, object discrimination, and proximity-based navigation, potentially limiting its transferability across diverse environments. The aerial camera method, while demonstrating greater robustness to parameter selection through reliance on categorical topological distinctions, introduces sensitivity to visual conditions including illumination, contrast, and shadow effects. These contrasting characteristics reflect the fundamental nature of each sensing modality: geometric precision through local proximity sensing versus global spatial reasoning through visual perception.

Several promising directions for future research emerge from this comparative study. First, investigating the impact of realistic sensor noise and state uncertainty through IMU-based measurements combined with Kalman filtering would reveal the robustness of each approach to localization errors under practical conditions. Second, the spatial accuracy limitations of the aerial camera approach could potentially be addressed through adaptive altitude control, where the aerial vehicle dynamically adjusts its operational height when approaching critical topological features. Third, extending the evaluation to environments containing passages narrower than vehicle dimensions would provide insights into traversability assessment capabilities. The range sensor approach would require reactive, trial-and-error exploration with physical navigation attempts to empirically determine passability, while the aerial camera's preemptive visual assessment could evaluate passage widths against vehicle geometry before committing to traversal—a distinction analogous to reactive workspace exploration versus deliberative configuration space reasoning. Finally, hybrid architectures that combine the complementary strengths of both sensing modalities merit investigation, potentially achieving high-precision topological mapping while maintaining the strategic planning advantages of visual perception.

In conclusion, the choice between range sensor-based and aerial camera-based exploration methodologies ultimately depends on mission-specific priorities and operational constraints. Applications requiring high-fidelity topological maps with precise metric accuracy would benefit from range sensor approaches despite temporal penalties, while time-critical exploration missions or energy-constrained platforms would find the operational efficiency of aerial camera systems more advantageous. This study provides quantitative insights and practical guidance for selecting appropriate exploration strategies based on application requirements, contributing to the broader understanding of autonomous exploration in unknown environments.

REFERENCES

[1] D. Dong, Z. Wang, J. Guan, Y. Xiao, and Y. Wang, "Research on key technology and application progress of rescue robot in nuclear accident emergency situation," Nuclear Engineering and Technology, vol. 57, no. 6, pp. 103457, 2025.

[2] K. You, C. Zhou, L. Ding, and Y. Wang, "Construction Robotics in Extreme Environments: From Earth to Space," Engineering, 2025.

[3] G. Fragapane, R. de Koster, F. Sgarbossa, and J. O. Strandhagen, "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda," European Journal of Operational Research, vol. 294, no. 2, pp. 405-426, 2021.

[4] Y. Bai, B. Zhang, N. Xu, J. Zhou, J. Shi, and Z. Diao, "Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review," Computers and Electronics in Agriculture, vol. 205, pp. 107584, 2023.

[5] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," Engineering Science and Technology, an International Journal, vol. 40, pp. 101343, 2023.

[6] K. N. McGuire, G. C. H. E. de Croon, and K. Tuyls, "A comparative study of bug algorithms for robot navigation," Robotics and Autonomous Systems, vol. 121, pp. 103261, 2019.

[7] L. Zhai, C. Liu, X. Zhang, and C. Wang, "Local Trajectory Planning for Obstacle Avoidance of Unmanned Tracked Vehicles Based on Artificial Potential Field Method," IEEE Access, vol. 12, pp. 19665-19681, 2024.

[8] A. Babinec, F. Duchoň, M. Dekan, P. Pásztó, and M. Kelemen, "VFHTDT (VFH with Time Dependent Tree): A new laser rangefinder based obstacle avoidance method designed for environment with non-static obstacles," Robotics and Autonomous Systems, vol. 62, no. 8, pp. 1098-1115, 2014.

[9] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: 'Follow the Gap Method'," Robotics and Autonomous Systems, vol. 60, no. 9, pp. 1123-1134, 2012.

[10] J. Ding, Y. Zhou, X. Huang, K. Song, S. Lu, and L. Wang, "An improved RRT* algorithm for robot path planning based on path expansion heuristic sampling," Journal of Computational Science, vol. 67, pp. 101937, 2023.

[11] S. Chen, G. Yang, G. Cui, S. Yi, and L. Wu, "Improved Path Planning and Controller Design Based on PRM," IEEE Access, vol. 13, pp. 44156-44168, 2025.

[12] A. A. N. Kumaar and S. Kochuvila, "Mobile Service Robot Path Planning Using Deep Reinforcement Learning," IEEE Access, vol. 11, pp. 100083-100096, 2023.

[13] Y. Li, R. Jin, X. Xu, Y. Qian, H. Wang, S. Xu, and Z. Wang, "A Mobile Robot Path Planning Algorithm Based on Improved A* Algorithm and Dynamic Window Approach," IEEE Access, vol. 10, pp. 57736-57747, 2022.

[14] H. Xi, W. Li, F. Zhao, L. Chen, and Y. Hu, "A Safe and Efficient Timed-Elastic-Band Planner for Unstructured Environments," in 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024, pp. 3092-3099.

[15] A. Joglekar, S. Sathe, N. Misurati, S. Srinivasan, M. J. Schmid, and V. Krovi, "Deep Reinforcement Learning Based Adaptation of Pure-Pursuit Path-Tracking Control for Skid-Steered Vehicles," IFAC-PapersOnLine, vol. 55, no. 37, pp. 400-407, 2022.

[16] L. Xu, C. Feng, V. R. Kamat, and C. C. Menassa, "An Occupancy Grid Mapping enhanced visual SLAM for real-time locating applications in indoor GPS-denied environments," Automation in Construction, vol. 104, pp. 230-245, 2019.

[17] J. Jiang, T. Zhang, and K. Li, "LiDAR-based 3D SLAM for autonomous navigation in stacked cage farming houses: An evaluation," Computers and Electronics in Agriculture, vol. 230, pp. 109885, 2025.

[18] W. Xue, R. Ying, Z. Gong, R. Miao, F. Wen, and P. Liu, "SLAM Based Topological Mapping and Navigation," in 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), 2020, pp. 1336-1341.

[19] J. Hou, Y. Yuan, and S. Schwertfeger, "Area Graph: Generation of Topological Maps using the Voronoi Diagram," in 2019 19th International Conference on Advanced Robotics (ICAR), 2019, pp. 509-515.

[20] P. G. C. N. Senarathne and D. Wang, "Incremental algorithms for Safe and Reachable Frontier Detection for robot exploration," Robotics and Autonomous Systems, vol. 72, pp. 189-206, 2015.

[21] M. Naazare, F. G. Rosas, and D. Schulz, "Online Next-Best-View Planner for 3D-Exploration and Inspection With a Mobile Manipulator Robot," IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 3779-3786, 2022.

[22] K. Pongsirijinda, Z. Cao, B. P. L. Lau, R. Liu, C. Yuen, and U.-X. Tan, "MEF-Explore: Communication-Constrained Multi-Robot Entropy-Field-Based Exploration," IEEE Transactions on Automation Science and Engineering, vol. 22, pp. 16062-16078, 2025.

[23] H. Kim, H. Kim, S. Lee, and H. Lee, "Autonomous Exploration in a Cluttered Environment for a Mobile Robot With 2D-Map Segmentation and Object Detection," IEEE Robotics and Automation Letters, vol. 7, no. 3, pp. 6343-6350, 2022.

[24] M. R. H. Al-Dahhan and K. W. Schmidt, "Voronoi Boundary Visibility for Efficient Path Planning," IEEE Access, vol. 8, pp. 134764-134781, 2020.

[25] B. B. K. Ayawli, A. Y. Appiah, I. K. Nti, F. Kyeremeh, and E. I. Ayawli, "Path planning for mobile robots using Morphological Dilation Voronoi Diagram Roadmap algorithm," Scientific African, vol. 12, pp. e00745, 2021.

[26] S.-K. Huang, W.-J. Wang, and C.-H. Sun, "A New Multirobot Path Planning With Priority Order Based on the Generalized Voronoi Diagram," IEEE Access, vol. 10, pp. 56564-56577, 2022.

[27] W. Chi, J. Wang, Z. Ding, G. Chen, and L. Sun, "A Reusable Generalized Voronoi Diagram-Based Feature Tree for Fast Robot Motion Planning in Trapped Environments," IEEE Sensors Journal, vol. 22, no. 18, pp. 17615-17624, 2022.

[28] W. Chi, Z. Ding, J. Wang, G. Chen, and L. Sun, "A Generalized Voronoi Diagram-Based Efficient Heuristic Path Planning Method for RRTs in Mobile Robots," IEEE Transactions on Industrial Electronics, vol. 69, no. 5, pp. 4926-4937, 2022.

[29] Y. Zhong, X. Zhu, J. Huang, Y. Ye, Z. Wang, and Y. Wang, "GVDF-RRT*: An Improved F-RRT* Path Planning Algorithm Based on Generalized Voronoi Diagram," IEEE Access, vol. 13, pp. 78968-78981, 2025.

[30] J. Wen, X. Zhang, Q. Bi, H. Liu, J. Yuan, and Y. Fang, "G²VD Planner: Efficient Motion Planning With Grid-Based Generalized Voronoi Diagrams," IEEE Transactions on Automation Science and Engineering, vol. 22, pp. 3743-3755, 2025.

[31] N. Kalra, D. Ferguson, and A. Stentz, "Incremental reconstruction of generalized Voronoi diagrams on grids," Robotics and Autonomous Systems, vol. 57, no. 2, pp. 123-128, 2009.

[32] A. Eldemiry, Y. Zou, Y. Li, C.-Y. Wen, and W. Chen, "Autonomous Exploration of Unknown Indoor Environments for High-Quality Mapping Using Feature-Based RGB-D SLAM," Sensors, vol. 22, no. 14, pp. 5117, Jul. 2022.

[33] D. Chen, X. Gong, A. Xiao, A. Luo, L. Sun, Y. Lv, J. Wang, and W. Chi, "GVD-Exploration: An Efficient Autonomous Robot Exploration Framework Based on Fast Generalized Voronoi Diagram Extraction," IEEE Transactions on Intelligent Vehicles, pp. 1-13, 2024.

[34] L. Li, X. Zuo, H. Peng, F. Yang, H. Zhu, D. Li, J. Liu, F. Su, Y. Liang, and G. Zhou, "Improving Autonomous Exploration Using Reduced Approximated Generalized Voronoi Graphs," Journal of Intelligent & Robotic Systems, vol. 99, no. 1, pp. 91-113, Jul. 2020.

[35] H. Zhao, Y. Guo, Y. Liu, and J. Jin, "Multirobot unknown environment exploration and obstacle avoidance based on a Voronoi diagram and reinforcement learning," Expert Systems with Applications, vol. 264, pp. 125900, 2025.

[36] Y. Lei, J. Hou, P. Ma, and M. Ma, "Voronoi-GRU-Based Multi-Robot Collaborative Exploration in Unknown Environments," Applied Sciences, vol. 15, no. 6, pp. 3313, 2025.

[37] M. Berg, O. Cheong, M. Kreveld, and M. Overmars, Computational Geometry: Algorithms and Applications, Berlin, Heidelberg: Springer, 2008.

[38] F. Aurenhammer, R. Klein, and D.-T. Lee, Voronoi Diagrams and Delaunay Triangulations, 1st ed., USA: World Scientific Publishing Co., Inc., 2013.

[39] R. V. Muratov, P. N. Ryabov, and S. A. Dyachkov, "Dynamic domain decomposition method based on weighted Voronoi diagrams," Computer Physics Communications, vol. 290, pp. 108790, 2023.

[40] J. Su, M. Beshley, K. Przystupa, O. Kochan, B. Rusyn, R. Stanisławski, O. Yaremko, M. Majka, H. Beshley, I. Demydov, J. Pyrih, and I. Kahalo, "5G multi-

tier radio access network planning based on voronoi diagram," Measurement, vol. 192, pp. 110814, 2022.

[41] M. Banach and R. Długosz, "Solutions for planning smart hybrid public transportation system based on Google Maps and Voronoi diagrams," Journal of Computational and Applied Mathematics, vol. 472, pp. 116775, 2026.

[42] M.-J. Kim, T. Y. Kang, and C.-K. Ryoo, "Real-Time Path Planning for Unmanned Aerial Vehicles Based on Compensated Voronoi Diagram," International Journal of Aeronautical and Space Sciences, vol. 26, no. 1, pp. 235-244, 2025.

[43] M. L. Gavrilova, Generalized Voronoi Diagram: A Geometry-Based Approach to Computational Intelligence, Berlin Heidelberg: Springer, 2008.

[44] W. Yu, H. K. Lee, S. Hariharan, W. Bu, and S. Ahmed, "Evolving generalized Voronoi diagrams for accurate cellular image segmentation," Cytometry Part A, vol. 77A, no. 4, pp. 379-386, 2010.

[45] L. Zhao, F. Guo, Y. Zhu, H. Wang, and B. Zhou, "A Generalized Voronoi Diagram-Based Segment-Point Cyclic Line Segment Matching Method for Stereo Satellite Images," Remote Sensing, vol. 16, no. 23, pp. 4395, 2024.

[46] I. Lee and C. Torpelund-Bruin, "Geographic knowledge discovery from Web Map segmentation through generalized Voronoi diagrams," Expert Systems with Applications, vol. 39, no. 10, pp. 9376-9388, 2012.

[47] X. Zhai and F. Chen, "Path Planning of a Type of Porous Structures for Additive Manufacturing," Computer-Aided Design, vol. 115, pp. 218-230, 2019.

CURRICULUM VITAE

SİBEL TUTAR

Education Information

Mersin University	Computer Engineering *(B.Sc.)*	2004-2008
Mersin University	Electrical and Electronics Engineering *(Minor in Telecommunications)*	2006-2008
Gaziantep University	Aeronautics and Aerospace Engineering *(M.Sc.)*	2023-2025

Work Experience

Abant Izzet Baysal University	Information Processing Center *Computer Programmer*	2010-2012
Gaziantep University	Information Processing Center *Computer Operator*	2012-2013
Gaziantep University	Department of Student Affairs *Computer Operator*	2013-2020
Gaziantep University	Distance Education Center *Computer Operator*	2020-2023
Gaziantep University	Institute of Science *Computer Operator*	2023-…