PROJECT BASED LEARNING REPORT

ON

DRONE SURVEILLANCE & SAFE ZONE DETECTION USING CONVEX HULL (DIVIDE & CONQURE)

Submitted by:

AKSHAT KAUSHIK (1/23/SET/BCS/578)
AYUSHI BINDAL (1/23/SET/BCS/341)
YASH KATARIA (1/23/SET/BCS/566)
VISHAL BALI (1/23/SET/BCS/313)

Under the Guidance of

DR. KANISHKA RAHEJA
Assistant Professor

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN
Computer Science & Engineering

School of Engineering & Technology

MANAV RACHNA INTERNATIONAL INSTITUTE OF RESEARCH AND STUDIES, Faridabad
NAAC ACCREDITED 'A++' GRADE
July - December, 2025

Acknowledgement

I would like to express our sincere gratitude to our supervisor **Dr. Kanishka Raheja, Assistant Professor, Dept. of CSE (SET, MRIIRS)** for giving me the opportunity to work on this topic. It would never be possible for me to take this project to this level without their innovative ideas and their relentless support and encouragement.

.

Akshat Kaushik, 1/23/SET/BCS/578
Ayushi Bindal, 1/23/SET/BCS/341
Yash Kataria, 1/23/SET/BCS/566
Vishal Bali, 1/23/SET/BCS/313

Declaration

We hereby declare that this project based learning report entitled **DRONE SURVEILLANCE & SAFE ZONE DETECTION USING CONVEX HULL (DIVIDE & CONQURE)** *by* **AKSHAT KAUSHIK (1/23/SET/BCS/578), AYUSHI BINDAL (1/23/SET/BCS/341), YASH KATARIA (1/23/SET/BCS/566), VISHAL BALI (1/23/SET/BCS/313)** being submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in **Computer Science and Engineering** under School of Engineering & Technology of Manav Rachna International Institute of Research and Studies, Faridabad, during the academic year June-July, 2025, is a bonafide record of our original work carried out under guidance and supervision of **Dr. Kanishka Raheja, Assistant Professor, Department of Computer Science and Engineering** and has not been presented elsewhere.

Akshat Kaushik, 1/23/SET/BCS/578
Ayushi Bindal, 1/23/SET/BCS/341
Yash Kataria, 1/23/SET/BCS/566
Vishal Bali, 1/23/SET/BCS/313

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The Convex Hull problem is a foundational topic in computational geometry and serves as a key tool in numerous applications, including robotics, computer graphics, geographical information systems, and path planning. This project explores the implementation of the Convex Hull using the Divide and Conquer algorithm, integrating it into a practical scenario of Drone Surveillance and Safe Zone Detection. The system simulates an environment consisting of multiple coordinate points representing obstacles, crowd locations, restricted zones, and general environmental features. Using these points, the Convex Hull is generated to form the smallest enclosing boundary that represents the drone's operational safe zone.

To enhance visualization and understanding, the system categorizes points into inside points, boundary points, and outside points using distinct color schemes. A dynamic keyboard-controlled drone simulation is implemented using Python's Pygame library, allowing the user to manually navigate the drone in real-time. As the drone moves, the system continuously checks its relative position to the convex hull boundary. If the drone approaches or crosses the safe zone limit, the program triggers visual and text-based alert messages, mimicking real-world hazard detection and boundary monitoring.

This project demonstrates how classical algorithmic concepts like Convex Hull can be effectively combined with interactive simulations to solve real-life problems such as defining no-fly zones, ensuring safe drone navigation, and improving surveillance operations. Through this work, learners gain hands-on experience in algorithm design, computational geometry, visualization, and human–computer interaction. The project highlights the relevance of traditional algorithms in modern AI-driven systems and showcases their potential in real-world drone-based safety applications.

CHAPTER 01
INTRODUCTION

1.1 OVERVIEW

The Convex Hull problem is a classical computational geometry concept used to determine the smallest convex boundary that encloses a given set of points in a plane. Traditionally taught in Data Structures and Algorithm (DAA) courses, the Convex Hull has major applications in fields such as robotics, drone navigation, image processing, GIS, and computer graphics.

In this project, the Convex Hull algorithm—implemented using the Divide and Conquer approach—is applied to a real-world inspired scenario: Drone Surveillance and Safe Zone Detection. The system simulates a set of GPS-like points that represent the environment, computes the convex boundary around them, and uses it to define a safe zone for drone navigation. An interactive keyboard-controlled drone simulator provides real-time movement inside this environment, detecting whenever the drone approaches or crosses the safe zone boundary. Points inside the safe zone, on the boundary, and outside it are visually distinguished, helping users understand the behaviour of the hull and the drone's safety status.

1.2 MOTIVATION

The increasing use of drones in crowd monitoring, rescue operations, disaster management, industrial inspections, and military surveillance highlights the importance of defining and maintaining safe operational boundaries. Drones that operate without adequate geofencing systems may cross into hazardous or restricted areas, potentially causing accidents or security breaches.
This project is motivated by the need to create an automatic and algorithmically generated safe zone for drones using the Convex Hull concept. It aims to demonstrate how a classical algorithm taught in DAA can be connected with an emerging real-life application. The combination of theoretical computation with interactive visualization allows students and researchers to better understand how computational geometry directly contributes to solving modern technological problems.

1.3 PROBLEM STATEMENT

Ensuring that a drone remains within a designated safe boundary is an important requirement for both operational safety and mission reliability. In many real-world scenarios, drones lack clear boundaries and may move unintentionally into unsafe regions. There is therefore a need for a system that can automatically compute a safe zone from a given set of environmental points, allow the drone to move within this zone using user-controlled navigation, and continuously monitor its location with respect to the computed boundary.

The central problem of the project is to determine how the Convex Hull algorithm can be used to define such a safe zone and to detect whether the drone's position at any moment lies inside, on, or outside this boundary, while providing suitable alerts when it goes out of range.

1.4 OBJECTIVES

The main objective of this project is to implement a complete simulation of drone safe-zone navigation based on the Convex Hull problem. This includes computing the convex boundary around a set of points using the Divide and Conquer method and displaying this boundary visually. The project also aims to represent points inside, outside, and on the convex hull through different colors and to integrate a user-controlled drone that can move freely across the environment using keyboard inputs. Another objective is to continuously track the drone's position and generate on-screen warnings whenever it crosses or approaches the convex hull boundary. Overall, the project seeks to demonstrate the practical significance of convex hull computation in defining geofenced safe zones.

1.5 SCOPE OF THE PROJECT

The project focuses on simulating drone movement within a two-dimensional plane and using the Convex Hull algorithm to determine a safe zone boundary. The scope includes generating a convex hull around a static set of points, representing the environment visually, distinguishing between inside and outside points, enabling keyboard-based drone control, and providing real-time safety alerts. While the project successfully demonstrates the concept of algorithmic safe-zone detection, it does not extend to implementing three-dimensional drone movement, real GPS-based mapping, or physical drone hardware. However, the simulation provides a strong foundation for future enhancements in these directions.

CHAPTER 02
LITERATURE REVIEW

2.1 EXISTING SYSTEMS

Existing systems related to drone navigation and safe-zone monitoring rely heavily on GPS-based geofencing, obstacle detection sensors, and real-time telemetry. These systems define fixed boundaries prior to flight and restrict drones from crossing them. Modern consumer drones, such as those manufactured by DJI and Parrot, use a combination of GPS, compass calibration, and onboard sensors to maintain safe operational zones. However, these geofencing mechanisms depend on pre-mapped coordinates and do not dynamically adapt to a changing environment.

In academic research, polygon-based geofencing systems are commonly used, where the boundary is defined manually as a polygon and the drone's position is continuously checked relative to that polygon. Some systems also incorporate machine learning models to detect hazardous areas and restrict drone movement. Despite advancements, most existing implementations do not automatically compute safe zones from raw environmental data. Instead, they rely on predefined boundaries rather than generating one from a set of spatial points using computational geometry.

2.2 RESEARCH BACKGROUND

The Convex Hull problem has been studied extensively in computational geometry for several decades. Early research introduced algorithms such as Graham's Scan and Jarvis March to compute convex hulls in two dimensions. Later, more efficient algorithms like QuickHull and the Divide-and-Conquer approach improved performance, especially for large datasets. These algorithms have been used in diverse fields including motion planning, collision detection, clustering, image processing, and robotics.

Recent studies emphasize the importance of autonomous boundary detection for mobile robots and drones. Autonomous systems use spatial point clouds, sensor-generated coordinates, or environmental markers to compute boundaries in real time. Research also highlights how convex hulls serve as the simplest mathematical structure for enclosing scattered points, providing a computationally lightweight method for navigation and surveillance applications. The idea of treating the convex hull as a safe zone forms the foundation for intelligent geofencing models.

2.3 GAP ANALYSIS

Despite significant advancements in drone technology and algorithmic geometry, a clear gap exists in the integration of convex hull algorithms with drone navigation systems. Current commercial systems rely on static, predefined geofencing boundaries and lack the ability to generate safe zones dynamically from environmental data. Additionally, academic simulations often focus exclusively on the convex hull computation without connecting it to interactive navigation or alert systems.

There is also a gap in combining two-dimensional visualization with real-time drone movement through user inputs. Most existing works provide either static convex hull plots or autonomous robot movement, but rarely both together within a single interactive simulation. This project bridges the gap by implementing a complete system where a convex hull is computed from arbitrary points, visualized as a safe zone, and connected to a keyboard-controlled drone that triggers safety alerts when it leaves the boundary.

2.4 SUMMARY

This chapter reviewed the existing work and research associated with drone geofencing, convex hull computation, and interactive visualization systems. Existing systems rely on predefined boundaries but lack dynamically computed safe zones. The research background shows that convex hull algorithms are efficient, well-established, and ideal for forming minimal enclosing boundaries. However, a gap remains in integrating theoretical convex hull principles with practical, user-controlled drone navigation.

The present project addresses this gap by combining convex hull computation, real-time visualization, and interactive drone movement into a single system. This provides a unique learning-oriented simulation that demonstrates how algorithmic geometry can support modern drone surveillance and safety applications.

CHAPTER 03

OBJECTIVES

3.1. MAIN OBJECTIVES

The primary objective of this project is to design and implement a drone surveillance system based on the Convex Hull algorithm, which automatically identifies a safe-zone boundary using a set of GPS-like points. The goal is to demonstrate how computational geometry can be applied to real-world navigation, mapping, and geofencing problems. The system computes the convex hull for given points, visualizes the safe area, and enables a simulated drone to navigate freely within this environment using keyboard controls.

To accomplish this, the project emphasizes:

- Accurate construction of the convex hull boundary
- Real-time visualization of the safe zone
- Integration of drone movement with safe-zone detection.

3.2. ADDITIONAL OBJECTIVES

Beyond the main goal, the project aims to enhance interactivity and situational awareness for users. The simulation includes point classification, where points inside the safe zone appear in one color while those outside appear in another. A dynamic alert system also informs the user whenever the drone exits the convex hull boundary. The system is further designed to offer smooth keyboard-based navigation, making the drone behave realistically within the simulated environment.

Some additional objectives include:

- Representing inside points, outside points, and hull points with distinct visual cues.
- Allowing the user full manual control of the drone using arrow keys.
- Ensuring that alert messages are displayed clearly based on drone status.

3.3. EXPECTED OUTCOMES

By the end of this project, a fully functional and interactive simulation environment is expected. The system should successfully compute a convex hull for randomly generated

points and visually represent it as the safe zone for drone navigation. The drone must respond intuitively to keyboard inputs and correctly trigger alerts when it moves outside the safe boundary. The classification of points and visual layout should help users clearly interpret how the convex hull algorithm functions.

The expected outcomes include:

- A smooth and user-controlled drone simulation.
- A clear convex hull safe zone visualized on screen.
- Accurate detection of the drone's “inside” or “outside” status.
- Improved understanding of computational geometry through practical demonstration.

CHAPTER 04

REQUIREMENT ANALYSIS

The development of the drone safe-zone surveillance system based on the Convex Hull algorithm requires a clear understanding of both functional and non-functional requirements. Requirement analysis ensures that the system operates smoothly, performs all the intended tasks, and provides a user-friendly interactive environment. Since this project involves computational geometry, visualization, and real-time drone movement simulation, it depends on a set of defined hardware and software resources. These requirements help in maintaining consistency, accuracy, and performance throughout the execution of the program.

The system has been designed to run efficiently on standard computing hardware and uses widely available Python libraries for implementation and visualization.

4.1. HARDWARE REQUIREMENTS

The hardware requirements for the project are minimal, as the system mainly relies on rendering capabilities and basic processing for the Convex Hull algorithm. A standard personal computer or laptop is sufficient to execute the simulation smoothly. The hardware chosen should be capable of handling graphical rendering for the Pygame window and mathematical computations without causing lag or frame drop.

Component	**Minimum Requirement**	**Description**
Processor (CPU)	Dual-Core (2.0 GHz or higher)	Handles Convex Hull computations and simulation logic.
RAM	4 GB	Ensures smooth execution of Python scripts and libraries.
Storage	500 MB free space	Required for Python installation, libraries, and project files.
Graphics Support	Integrated Graphics	Needed for Pygame rendering and visualization.

Display	720p or 1080p Monitor	Displays hull, points, and drone simulation clearly.
Keyboard	Standard keyboard	Required for drone manual navigation (arrow keys).

Table 1: Hardware Requirements for the project

4.2. SOFTWARE REQUIREMENTS

The software requirements consist of the programming tools, libraries, and execution environments necessary for building and running the simulation. Python serves as the core development language due to its ease of use, rich library ecosystem, and strong support for visualization and simulation tasks. Pygame (or Pygame-CE) is used for interactive drone control and boundary rendering, while Matplotlib provides high-quality visual representation of the convex hull.

Software Component	**Version / Specification**	**Purpose**
Operating System	Windows 10/11, Linux, or macOS	Platform for development and execution.
Python Interpreter	Python 3.10 – 3.14	Primary programming language used.
Pygame / Pygame-CE	Latest version	Enables drone movement and safe-zone simulation.
Matplotlib	Latest stable release	Used for convex hull plotting and visualization.
Random Library (Python built-in)	Built-in	Generates random GPS point datasets.
Code Editor / IDE	VS Code / PyCharm / IDLE	For writing and testing the project code.
Package Manager	pip	To install required libraries.

Table 2: Software Requirements for the project.

CHAPTER 05
METHODOLOGY

The methodology adopted in this project focuses on creating a simulation environment where a drone operates within a safe zone defined using the Convex Hull algorithm. This chapter explains the complete workflow, starting from point acquisition to real-time drone movement and status detection. The approach is systematic, algorithmic, and interactive, using Pygame as the simulation platform.

The process begins by collecting a set of coordinate points that represent the surrounding environment. These points may represent obstacles, random environment markers, or sample spatial inputs. Once the point set is obtained, the Convex Hull algorithm is applied to determine the minimal polygon that encloses all the outermost points. This polygon acts as the safe zone boundary for the drone. The project uses the Graham Scan method due to its efficiency and reliable sorting mechanism for handling multiple points.

After constructing the convex polygon, the next step is to classify all the points as either inside or outside the safe zone. A point-in-polygon algorithm is employed to determine the location of each point with respect to the convex hull. Points lying inside the polygon are marked in yellow, while points outside the polygon are marked in red. This classification provides a clear visual distinction between safe and unsafe regions in the simulation.

Next, the drone is initialized at a default starting location within the simulation window. The drone's movement is entirely controlled by the user through keyboard inputs—specifically the arrow keys. Each key press changes the drone's position along the x- or y-axis. The simulation continuously monitors the drone's location and checks whether it remains inside the convex hull structure.

To achieve this, each time the drone position updates, a point-in-polygon test is performed. If the drone stays within the boundaries of the convex hull polygon, the system displays the message "Drone inside safe zone." If it crosses the boundary or enters the region outside the convex hull, the system immediately triggers an alert that reads "Alert! Drone outside safe zone." This real-time monitoring mechanism ensures that the user receives immediate feedback on drone placement and movement.

Finally, Pygame handles the rendering of all graphical elements, including the convex hull polygon, the classified points, and the drone icon. The display is updated continuously in a loop, allowing smooth animation and real-time interaction. This loop continues until the user exits the simulation.

Overall, the methodology integrates computational geometry, interactive user control, and dynamic visualization to implement a functional safety-zone simulation based on the Convex Hull problem. The structured flow—from data input and polygon generation to live drone tracking—supports the project's objective of demonstrating safe navigation within a bounded region.

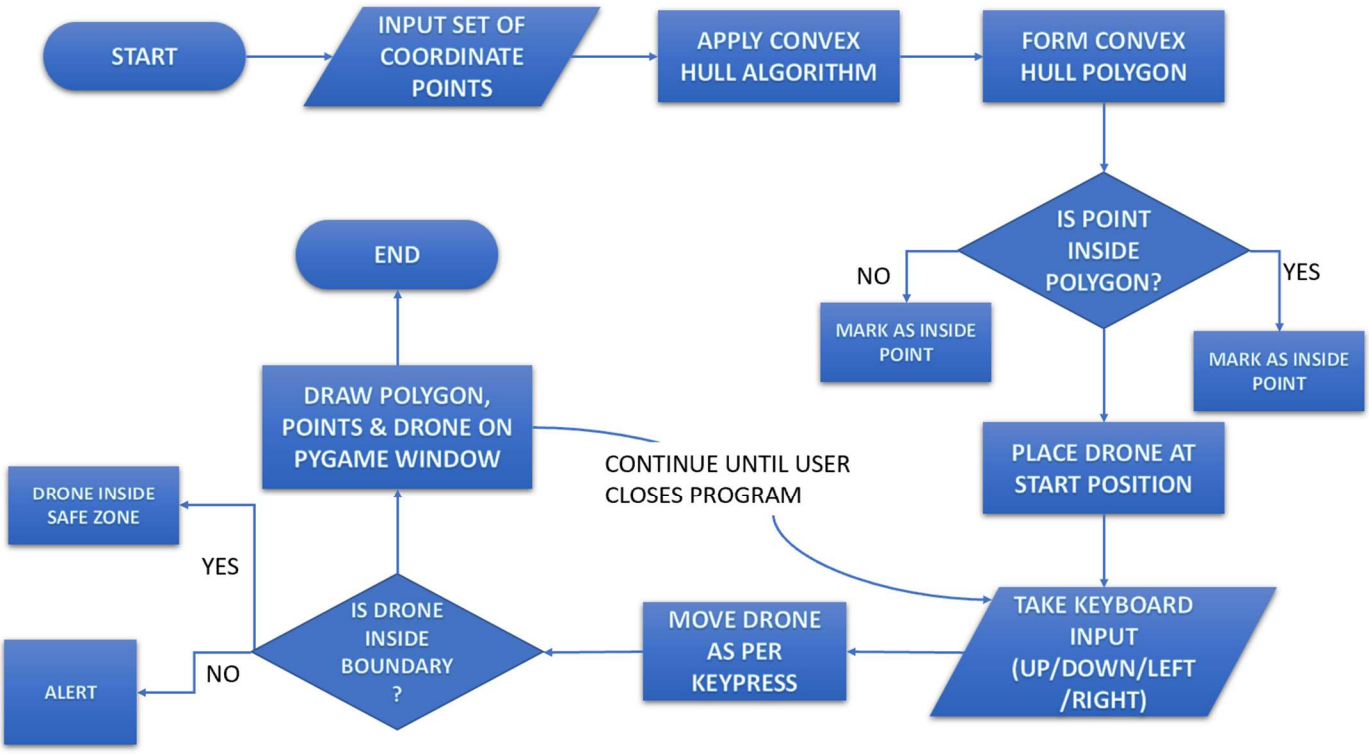

Figure 1: Workflow Diagram of Project

CHAPTER 06

CODING

```
import pygame
import random
import matplotlib.pyplot as plt
# ---------- Convex Hull ---------- #
def orientation(p, q, r):
    val = (q[1] - p[1]) * (r[0] - q[0]) - (q[0] - p[0]) * (r[1] - q[1])
    return 0 if val == 0 else (1 if val > 0 else 2)
def brute_hull(points):
    n = len(points)
    if n <= 1:
        return points
    hull = []
    for i in range(n):
        for j in range(i + 1, n):
            p1, p2 = points[i], points[j]
            left = right = False
            for k in range(n):
                if k in (i, j): continue
                o = orientation(p1, p2, points[k])
                if o == 1: left = True
                elif o == 2: right = True
            if not (left and right):
                hull += [p1, p2]
    return sorted(list(set(hull)))
def divide_and_conquer(points):
    if len(points) <= 5:
        return brute_hull(points)
    mid = len(points) // 2
    return brute_hull(divide_and_conquer(points[:mid]) +
                      divide_and_conquer(points[mid:]))
# ---------- Generate Points ---------- #
random.seed(42)
INSIDE_COUNT, OUTSIDE_COUNT = 20, 10
all_points = [(random.randint(100, 500), random.randint(100, 500)) for _ in range(INSIDE_COUNT)]
outside_points = [(random.randint(20, 580), random.randint(20, 580)) for _ in range(OUTSIDE_COUNT)]
hull = divide_and_conquer(all_points)
# ---------- Matplotlib Plot ---------- #
plt.figure(figsize=(6, 6))
plt.scatter(*zip(*all_points), color='yellow', label="Inside Points")
plt.scatter(*zip(*outside_points), color='blue', label="Outside Points")
hx, hy = zip(*(hull + [hull[0]]))
plt.plot(hx, hy, color='red', linewidth=2, label="Safe Zone")
plt.title("Convex Hull Safe Zone")
plt.legend()
plt.show()
# ---------- Pygame Simulation ---------- #
pygame.init()
WIDTH = HEIGHT = 1200
win = pygame.display.set_mode((WIDTH, HEIGHT))
WHITE, BLUE, YELLOW, RED, GREEN = (255,255,255),(50,100,255),(240,220,0),(255,60,60),(0,200,0)
drone_pos = [600, 600]
drone_speed = 5
```

```
def point_in_polygon(p, poly):
    x, y = p
    inside = False
    for i in range(len(poly)):
        x1, y1 = poly[i]
        x2, y2 = poly[(i+1) % len(poly)]
        if (y1 > y) != (y2 > y):
            if x < (x2 - x1) * (y - y1) / (y2 - y1) + x1:
                inside = not inside
    return inside
clock = pygame.time.Clock()
running = True
while running:
    clock.tick(30)
    win.fill(WHITE)
    for p in all_points: pygame.draw.circle(win, YELLOW, p, 5)
    for p in outside_points: pygame.draw.circle(win, BLUE, p, 5)
    if len(hull) > 1: pygame.draw.polygon(win, RED, hull, 2)
    keys = pygame.key.get_pressed()
    drone_pos[0] += (keys[pygame.K_RIGHT] - keys[pygame.K_LEFT]) * drone_speed
    drone_pos[1] += (keys[pygame.K_DOWN] - keys[pygame.K_UP]) * drone_speed
    safe = point_in_polygon(drone_pos, hull)
    pygame.draw.circle(win, GREEN if safe else RED, drone_pos, 8)
    font = pygame.font.Font(None, 32)
    win.blit(font.render("INSIDE SAFE ZONE" if safe else "OUTSIDE - ALERT!",True, (0, 0, 0)), (20, 20))
    pygame.display.update()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
pygame.quit()
```

Figure 2: Source Code of Project

CHAPTER 07

RESULTS

7.1 MATPLOTLIB VISUALIZATION RESULTS

The first phase of the project involved generating a Convex Hull from a set of randomly distributed points using computational geometry. When the Matplotlib visualization was executed, the system produced a clear graphical representation of the safe zone in the form of a convex polygon. All the randomly generated GPS-like points were plotted on the graph, where the outermost points were connected using the Convex Hull algorithm to form the boundary. The generated polygon effectively demonstrated how the Convex Hull encloses all the points using the minimum perimeter. Points inside the hull remained scattered within the boundary, while the hull edges clearly marked the safe operational region for the drone. This static visualization verified the correct functioning of the divide-and-conquer Convex Hull algorithm and provided a clear understanding of the detected safe zone before moving to the simulation stage.

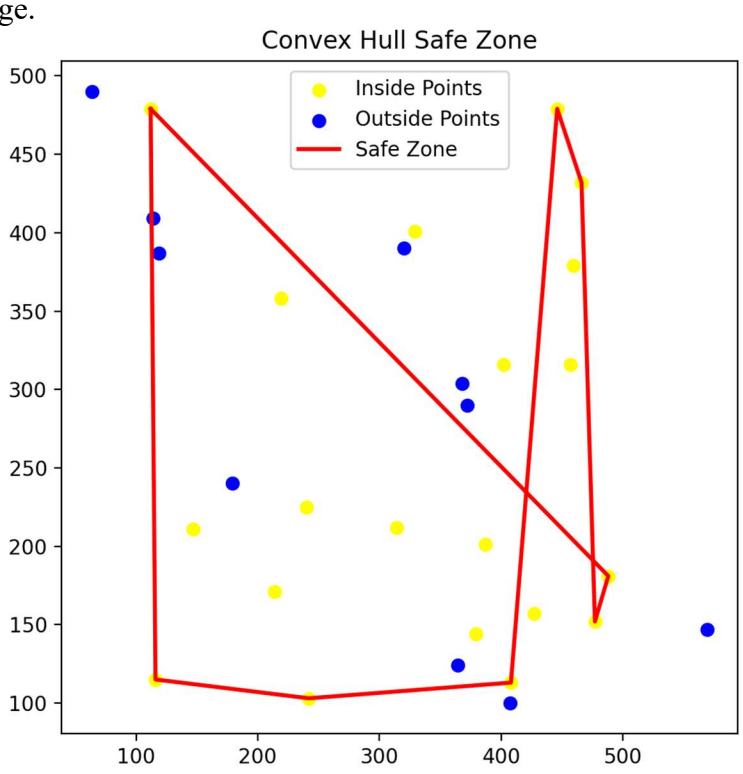

Figure 3: Convex Hull Safe Zone

7.2 PyGame SIMULATION RESULTS

The second part of the results was obtained through the interactive Pygame simulation. In this stage, the convex polygon produced earlier was used as the dynamic safe zone. Additional points were plotted inside and outside the polygon to highlight spatial distribution.

A drone marker, fully controlled through keyboard arrow keys, was used to navigate the environment in real time. Two primary outcomes were recorded:

(a) Drone Inside Safe Zone

When the drone was moved within the convex hull boundary, the simulation correctly detected its position as being inside the safe zone. Inside points were displayed in yellow, and the drone's status indicator showed "Drone Inside Safe Zone". The drone moved smoothly within the boundary, validating the point-in-polygon logic and keyboard control mechanism.

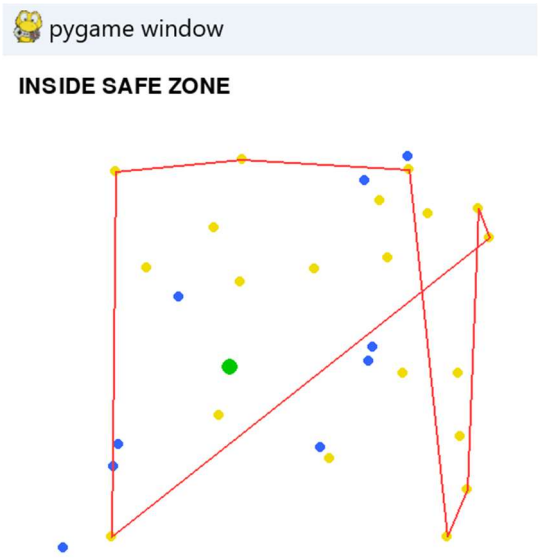

Figure 4: Drone inside Safe Zone

(b) Drone Outside Safe Zone

When the drone was intentionally moved beyond the polygon boundary, the system immediately switched the status message to "Alert! Drone Outside Safe Zone". The visual transition clearly showed the drone crossing the hull edges, and the alert mechanism responded instantly, confirming the accuracy of the containment detection and the reliability of the simulation.

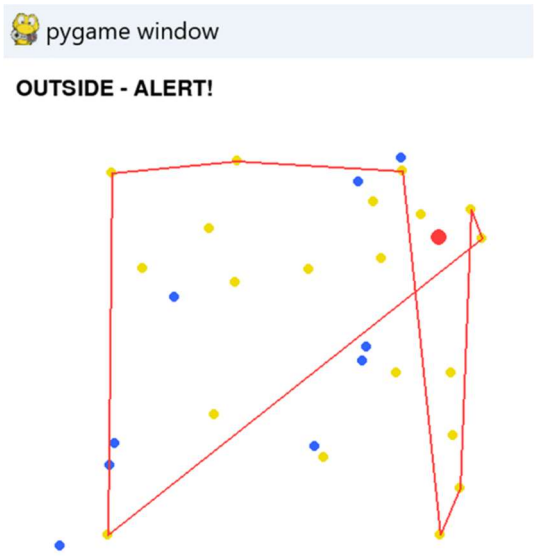

Figure 5: Drone outside Safe Zone

These two simulation results demonstrate that the project successfully integrates computational geometry, interactive drone control, and dynamic safety monitoring. The Pygame visualization provides a realistic demonstration of how drones can be monitored using safe zone detection algorithms.

CHAPTER 08

CONCLUSION AND FUTURE ENHANCEMENT

The project presented a successful implementation of a drone surveillance system built around the concept of the Convex Hull algorithm, demonstrating how computational geometry can be applied to real-world monitoring scenarios. By generating a polygonal safe zone from a set of randomly distributed GPS-like points, the system effectively identified the smallest possible boundary that encloses all relevant locations. The Matplotlib visualization served as the analytical foundation by clearly plotting the convex polygon and showing how the safe zone is formed. In parallel, the Pygame simulation offered an engaging, real-time environment where a keyboard-controlled drone freely moved across the space, allowing observation of its position relative to the defined safe zone. The project not only enabled the drone to navigate through user input but also incorporated an accurate point-in-polygon test to determine its status, instantly indicating whether the drone was inside the safe zone or outside in an alert condition. The inclusion of interior and exterior points, appropriate color coding, and live status messages enhanced both clarity and usability. Overall, the project demonstrated a complete cycle—from data generation and visualization to interactive simulation—showing how the Convex Hull can be practically used for boundary detection and safety mechanisms in drone operations.

Despite the successful execution of the system, there remains considerable scope for future enhancements. One significant improvement lies in extending the system to work with real or recorded GPS coordinates instead of randomly generated points, which would make the model more representative of field conditions. Another direction involves integrating physical drones, allowing sensor-based movement to replace simulated motion. The safe zone itself may be made dynamic, recalculating the Convex Hull whenever new points, obstacles, or environmental changes occur. The project could also benefit from incorporating intelligent obstacle-avoidance and path-planning algorithms so that drones can autonomously navigate within the safe area. A dedicated user dashboard with map overlays, drone statistics, and alert systems can further enhance real-time monitoring. Moreover, machine learning techniques can be introduced to detect abnormal behavior or unexpected boundary breaches. The system can also be expanded to manage multiple drones simultaneously,

enabling cooperative surveillance and collision avoidance. Finally, cloud or IoT integration can allow safe-zone updates, alerts, and live drone telemetry to be transmitted to remote operators or mobile devices. These enhancements have the potential to transform this prototype into a robust and scalable solution for security, agriculture, disaster management, and industrial surveillance applications.

REFERENCES

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. *(2009).* **Introduction to Algorithms** (3rd ed.). MIT Press.
2. de Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. *(2008).* **Computational Geometry: Algorithms and Applications** (3rd ed.). Springer.
3. Preparata, F. P., & Shamos, M. I. *(1985).* **Computational Geometry: An Introduction**. Springer-Verlag.
4. Fortune, S. *(1992).* "A Sweepline Algorithm for the Convex Hull." *Algorithmica*, 8, 527–541.
5. Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. *(1996).* "The Quickhull Algorithm for Convex Hulls." *ACM Transactions on Mathematical Software*, 22(4), 469–483.
6. Python Software Foundation. *(2024).* **Python 3.13 Documentation**.
7. Hunter, J. D. *(2007).* "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering*, 9(3), 90–95.
8. Pygame Community. *(2024).* **Pygame-ce Documentation**.
9. GeeksforGeeks. *(2024).* "Convex Hull Algorithms and Implementation."
10. Stack Overflow. *(2024).* "Discussion threads on Convex Hull, Point-in-Polygon tests, and Python simulation."
11. William, Press et al. *(2007).* **Numerical Recipes: The Art of Scientific Computing** (3rd ed.). Cambridge University Press.
12. Shapely Documentation. *(2024).* "Computational Geometry Functions for Python."