
# A Swarm Robotics Navigation Simulator for Phenotyping Soybean Plants using Voronoi-Ant Colony Optimization

**Sainath Reddy Gummi[1], Mohammad Ashik Alahe[1], Chulwoo Pack[2],**

**Young Chang[1]***

[1]Department of Agricultural and Biosystems Engineering, South Dakota State University, Brookings, SD 57007, USA

[2]Department of Electrical Engineering and Computer Science, South Dakota State University, Brookings, South Dakota, USA.

*Corresponding Author

**A paper from the Proceedings of the**

**16th International Conference on Precision Agriculture**

**21-24 July 2024**

**Manhattan, Kansas, United States**

***Abstract***

*Swarm intelligence, inspired by the collective behavior of social insects, plays a pivotal role in evolutionary computing and multi-agent systems. In agricultural contexts, swarm intelligence-inspired robotics can significantly enhance early detection of lower foliar diseases, which is essential for sustaining food production and economic viability over the long term. Soybean crops are susceptible to diseases like Sclerotinia stem rot, causing substantial economic losses to farmers. Traditional methods, including satellite and drone technologies, are insufficient for managing rapid disease spread due to cloud cover and limited field scalability. This study addresses these challenges by introducing the Voronoi-Ant Colony Optimization (V-ACO) framework, a hybrid algorithm combining Voronoi tessellation and Ant Colony Optimization (ACO) to enhance swarm robots' navigation and target identification. The V-ACO framework leverages the decentralized decision-making capabilities of swarm robotics, augmented by the path optimization features of ACO and spatial segmentation using Voronoi diagrams. This integration allows for efficient navigation and task performance in complex agricultural environments.*

*The framework was implemented and tested using the Unity engine, providing a scalable and realistic 3D simulation environment ideal for large-scale agricultural applications. The performance analysis of the V-ACO framework demonstrates its strengths and potential in target plant identification by following collision avoidance between dense plant spacing crops such as soybean in Unity 3D simulation. The results highlight the effective spatial distribution of detections*

*within Voronoi zones and emphasize the importance of advanced path planning using swarm robotics. Future work will focus on integrating deep computer vision models using Unity's ML-Agents to enhance the framework's capabilities further. Additionally, field trials using a swarm of robots will validate the framework's efficacy in real crop situations, providing valuable insights and guiding further optimizations to ensure its practical applicability and effectiveness in precision agriculture. The potential of this project lies in performing research using 3D robotics simulators to identify efficient path planning algorithms for swarm robotics navigation in crops with dense plant spacing.*

***Keywords***
*Swarm Robotics, Precision Agriculture, Voronoi diagrams, Ant Colony Optimization, Path Planning, Soybean, Unity*

# 1 Introduction

Swarm intelligence is a pivotal concept in evolutionary computing. The essential idea of swarm intelligence algorithms is to employ many simple agents applying almost no rule which in turn leads to an emergent global behavior. The general principles of swarm intelligence include the proximity principle, ensuring direct behavioral responses among agents. The quality principle, where agents respond to quality factors in their environment. The principle of diverse response, equipping agents to handle scattered resources and environmental fluctuations. Lastly, the principle of stability and adaptability, enabling agents to adapt to environmental fluctuations with minimal energy costs for mode changes (Hu, 2012; Hamann, 2018; Dias et al., 2021).

In terms of specific algorithms, the Ant Colony Optimization (ACO) algorithm mimics the foraging behavior of ants, particularly their use of chemical trails to find the shortest path. This algorithm finds applications in vehicle routing problems, image processing, assignment problems, and scheduling problems. Similarly, Bee Colony Optimization (BCO) algorithm is computed based on the foraging behavior of honeybees, specifically the waggle dance used to communicate distance, direction, and quality of food sources. Applications of these algorithms include shop scheduling, neural network training, and image processing. Another noteworthy algorithm is Particle Swarm Optimization (PSO), where agents move together to achieve a common goal, mimicking physical quantities such as velocity and position. The challenge in PSO lies in finding the optimal path with fewer parameters, with applications in finding global optimal solutions (Hu, 2012). Swarm robotics applies these principles to multi-agent systems, emphasizing decentralized control, complex communication protocols, and emergent behavior. Key attributes of swarm robotics include robustness, flexibility, and scalability. Robustness ensures that the system can perform target tasks with independent agents. Flexibility allows agents to serve multiple purposes and tasks, functioning under restricted communication. Scalability ensures that the system remains functional with varying numbers of elements, ensuring task completion (Hamann, 2018; Dias et al., 2021, Zhou et al., 2022).

## 1.1 Why Swarm Intelligence in Agriculture?

In agricultural contexts, swarm intelligence holds significant potential. For instance, the soybean crop is susceptible to various diseases which burdens farmers economic situation. The soybean Sclerotinia stem rot is caused by *Sclerotinia sclerotiorum*. This is one of the major fungal diseases in the north-central region of the United States. Between 2018 and 2022, Sclerotinia stem rot alone caused a loss of 92,804,866 bushels, translating to an economic loss of $990,455,168 in Northern U.S. States (Crop Protection Network, 2023). This disease affects the lower portion of plant stems and leaves and slowly spreading toward the upper portion. One approach for managing Sclerotinia stem rot is by spraying fungicides. However, spot spraying with drone at first mission might not be effective in detecting infected plants early enough for effective intervention. If ground robots can identify these infected plants and take early preventive measures such as fungicide application, it can significantly enhance disease management. Vinclozolin, for instance, has been shown to be highly effective in inhibiting *S. sclerotiorum*

mycelial growth at early stages (Mueller et al., 2002). Designing crop-specific robots that navigate between the rows can detect the early infected plants under the canopy. Subsequently, sending drones to these areas would be efficient for targeted fungicide spraying. This highlights the potential of ground swarm robotics in Precision Agriculture by enabling targeted fungicide applications.

To achieve collective intelligence in swarm robots, several components are integrated. The primary Central Processing Unit (CPU) coordinates swarm behavior, executes high-level algorithms, and makes decisions based on collective information. Auxiliary Micro Controller Unit (MCU) modules manage time-sensitive operations, ensuring timely responses and coordination within the swarm. Sensors and transducers perceive surroundings and gather information, crucial for collective intelligence relying on inter-robot communication. Actuators and transducers, including DC motors, gripper motors, LED lights, and speakers, interact with the environment and neighboring robots (Albiero et al., 2022; Dias et al., 2021). However, implementing swarm robotics in agriculture faces several challenges. These include ensuring hardware upgrades, maintaining robust communication networks, navigating complex field geometries, and developing control architectures to facilitate emergent group behaviors (Albiero et al., 2022). Simulation studies play a critical role in addressing these challenges by providing a controlled environment to test and refine algorithms, hardware configurations, and communication protocols before deployment in real-world settings.

### 1.2 Swarm Robotics Simulators for crop row navigation

Simulations allow researchers to model complex agricultural environments, optimize robot interactions, and predict system behaviors under various scenarios, reducing the risks and costs associated with field trials. Based on a comparison of Unity, Gazebo, and V-rep, Unity emerges as the most suitable simulator for large-scale agricultural applications (Wang et al., 2024). Unity offers high scalability, supporting simulations of very large environments making it ideal for realistic 3D simulations. Gazebo, while open-source and providing high-reliability simulations with excellent Robot Operating System (ROS) integration, is limited in scalability and might not be typically used for fields over a quarter section (160 acres). V-rep, known for its advanced simulation features and good ROS integration, also falls short in handling large fields. Therefore, for agricultural research requiring extensive field simulations, Unity is the preferred choice due to its scalability and robust 3D simulation capabilities (Table 1). The choice of simulation platforms like Unity further enhances the potential for realistic and scalable agricultural research, paving the way for innovative applications and improved farm management (Lim et al., 2021; De Melo et al., 2019; Karunarathna et al., 2023).

**Table 1. Swarm robotics simulators for crop row navigation**

| Simulator | 3D/2D | ROS integration | Suitable for simulating Quarter section Field |
|---|---|---|---|
| **Unity** | 3D | Yes | Yes |
| **Gazebo** | 3D | Yes | No |
| **V-rep** | 3D | Yes | No |

### 1.3 V-ACO: The hybrid algorithm approach

Swarm intelligence algorithms for soybean crop row navigation can be significantly enhanced using the Voronoi-Ant Colony Optimization (V-ACO) framework. This hybrid approach integrates swarm robotics (SR), Voronoi tessellation, and Ant Colony Optimization (ACO) to navigate and perform tasks efficiently in a simulated soybean farm environment (Dorigo & Stutzle, 2004; Okabe et al., 2000). The V-ACO framework leverages the decentralized decision-making capabilities of SR, enhanced by the path optimization features of ACO, guided by spatial segmentation using Voronoi diagrams (Du et al., 1999; Dorigo et al., 2006; Xiong et al., 2019). In our framework, Voronoi diagrams are used to partition the soybean field into regions based on the positions of robots or targets. This spatial segmentation allows for efficient allocation of tasks and navigation paths. ACO, inspired by the foraging behavior of ants, is employed to find the shortest path or

optimal sequence of actions within these regions. The V-ACO framework has been implemented and tested using the Unity engine, which provides a scalable and realistic 3D simulation environment ideal for large-scale agricultural applications.

# 2 Methods

## 2.1 Layout and design of the Simulated Soybean Field and Robot model

The simulation environment for the V-ACO framework was developed in Unity software version 2022.3.29f1. The development and testing of the simulation were conducted on a desktop computer equipped with AMD Ryzen 7 5800X processor, 32 GB of DDR4 RAM, and an NVIDIA GeForce RTX 3090 Graphics Processing Unit (GPU) with 24 GB of VRAM. This setup provided the necessary computational power to handle 3D simulation scenes and process iterative tasks efficiently for a 40 m × 40 m field.

The field was designed to reflect realistic crop recommendations on soybean planting (South Dakota State University Extension, 2020). Each row of soybean plants was spaced 0.762 meters apart (30 inches), while individual plants within a row were spaced at intervals of 0.1016 meters (4 inches). A 1-meter margin was maintained around the field's perimeter to avoid boundary effects. In total, 18,750 plants were instantiated in the field. Of these, 14,765 (80%) were healthy plants, and 3,985 (20%) were diseased plants. The distribution of soybean plants within the field was randomized to simulate natural planting variations (Table 2). Healthy plants used a specific prefab, while non-healthy plants used a different prefab, ensuring visual distinction in the simulation (Appendix 1, Fig. S1). The visualization in Figs. 1B & 1C depicts the uniform row spacing and top view of 40 m × 40 m simulated field across the simulation.

**Table 2. Simulation and Real-World Measurements for Soybean Field Design**

| Aspect | Unity Measurement | Real World Measurement (meters) | Real World Measurement (feet/inches) |
|---|---|---|---|
| **Field Width** | 40 units | 40 meters | 131.234 feet |
| **Field Length** | 40 units | 40 meters | 131.234 feet |
| **Field Area** | - | 1600 meters$^2$ | 0.395 acre |
| **Row Spacing** | 0.762 units | 0.762 meters | 30 inches |
| **Plant Spacing** | 0.1016 units | 0.1016 meters | 4 inches |
| **Margin** | 1 unit | 1 meter | 3.281 feet |
| **Total Number of Plants** | 18,750 | - | - |
| **Healthy Plants** | 14,765 (80%) | - | - |
| **Non-Healthy Plants** | 3,985 (20%) | - | - |

Four robot models were introduced into the simulation (Fig. 1A), each representing a real-world counterpart with dimensions suitable for navigating soybean fields. Specifically, each robot model had a width of 18 inches, aligning with the 0.762 meters (30 inches) row spacing between soybean plants. The robot prefab was instantiated based on real-world sizes and settings, as detailed in Appendix 2, Fig. S2. The Unity NavMesh, depicted in Fig. 1D, was generated using Unity's AI engine and was adjusted to accommodate the size of the rows and the robots (Unity Technologies, 2022). The NavMesh agent settings included a 0.4572 meters (18 inches) agent radius, ensuring precise navigation and interaction within the field environment. This setup allowed the robots to move seamlessly between the rows, effectively simulating real-world

conditions and validating the robot's ability to navigate and perform tasks in a controlled, virtual soybean farm. The detailed specifications and configurations of the robot prefab and NavMesh are crucial for ensuring the accuracy and reliability of the simulation results, which is described in Appendix 2, Table S1.

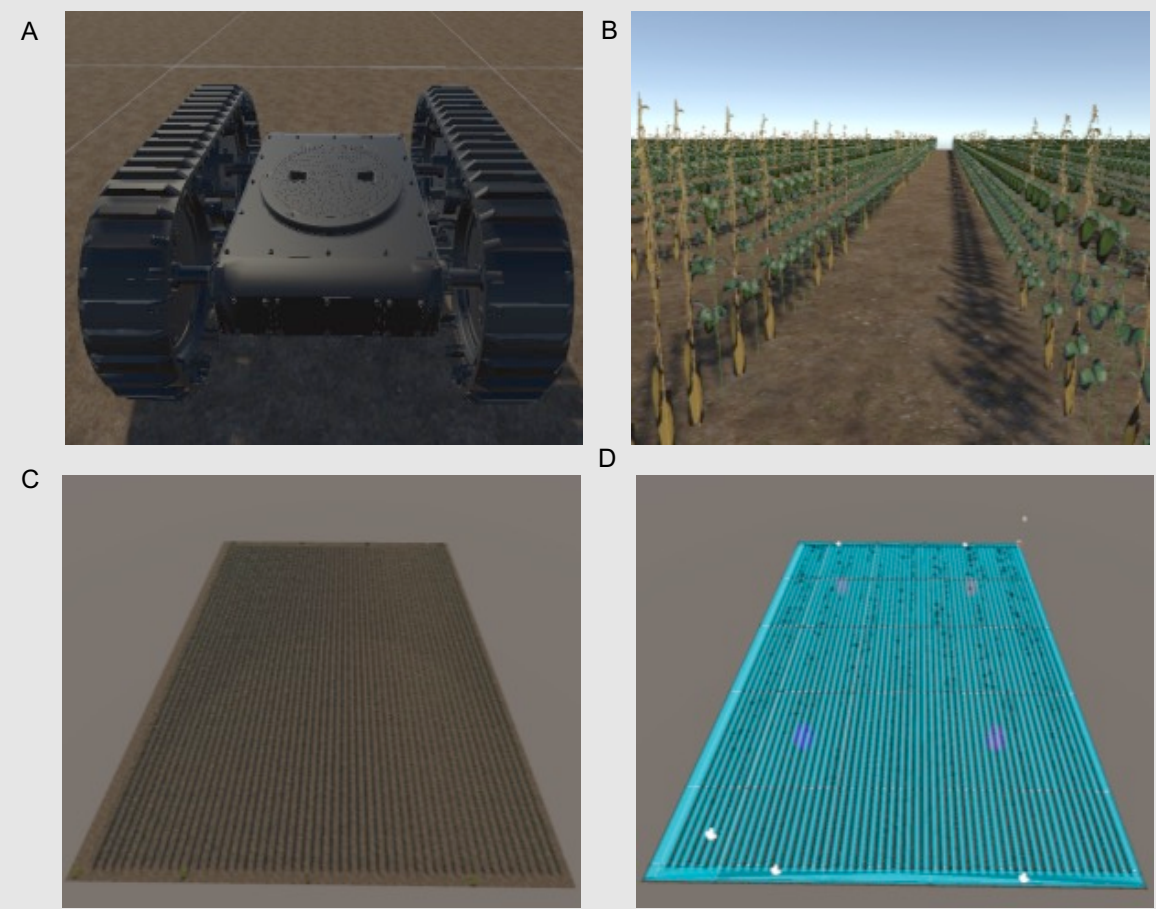


**Fig 1. Simulation visualization setup for testing V-ACO algorithm. (A) The digital robot model. (B) The row spacing visualized in the simulation. (C) Top view of the farm after initializing 40 m x 40 m planting. (D) NavMesh generated using Unity AI engine, adjusted to row size and robot size.**

## 2.2 V-ACO Framework for target plant identification

In the context of soybean diseased prefab plant identification, the ACO algorithm optimizes the paths of robots as they navigate through the Voronoi cells. The pheromone update rule in ACO influences the probability of path selection by subsequent robots, ensuring that optimal paths are reinforced over time (Dorigo & Stutzle, 2004; Okabe et al., 2000). This process involves pheromone evaporation to avoid premature convergence and the deposition of new pheromones to mark efficient paths (Şahin, 2005; Yang, 2014). The probability of selecting a particular path is influenced by the amount of pheromone present and the visibility, or attractiveness, of the path. This comprehensive integration of Voronoi tessellation and ACO within the V-ACO framework allows for effective navigation and task performance in complex agricultural environments. This framework was tested in simulation environments to evaluate the extent of V-ACO's effectiveness. The implementation details and algorithms described in this section are based on the custom C# scripts developed for this study, utilizing Unity's game engine for simulation and SciPy for computational geometry tasks (Unity Technologies, n.d.; Virtanen et al., 2020).

### *2.2.1 Voronoi Zone Management*

The Voronoi Zone Manager class is defined in the C# script, which divides the simulation into zones based on the location of each robot, ensuring coverage and optimizing navigation paths.

**Algorithm 1** Voronoi Zone Initialization

1: **Input**: Set of robot positions $R$, Field dimensions $F$

2: **Output**: Voronoi zones $V$

3: Initialize Voronoi zones $V$ assigned based on placement of robot positions $R$

4: **for** each point $p$ in the field $F$ **do**

5: Assign $p$ to the nearest robot zone in $V$

6: **end for**

*2.2.2 Robot Navigation Control*

The Robot Controller class is defined in the C# script, which manages the movements of each robot within its assigned Voronoi zone, utilizing local information and pheromone trails to make decisions.

**Algorithm 2** Robot Navigation

1: **Input**: Current position $P$, Target positions $T$, Pheromone map $M$

2: **Output**: Next position $P_{next}$

3: **if** target detected within the zone, **then**

4: Move directly towards target

5: **else**

6: Use pheromone levels to choose next position

7: Update position based on shortest path to next high pheromone concentration

8: **end if**

*2.2.3 Pheromone Mapping*

The Pheromone Map class is defined in the C# script to simulate the laying and evaporation of pheromones within the environment, guiding the robots' movement decisions.

**Algorithm 3** Pheromone Update

1: **Input**: Robot positions $R$, Detected targets $D$

2: **Output**: Updated pheromone concentrations M

3: **for** each robot $r$ in $R$ **do**

4: Lay pheromones along path $r$

5: **end for**

6: Evaporate pheromones over time

*2.2.4 Diseased Plant Detection*

Robots are equipped with RGB cameras to detect tagged diseased plants. When a diseased plant is detected, its position is logged, and a high concentration of pheromones is laid to attract other

robots for further investigation.

**Algorithm 4** Diseased Plant Detection

1: **Input**: Current position *P*, Sensor readings *S*

2: **Output**: Detection log *L*, Updated pheromone map *M*

3: **if** sensor detects diseased plant **then**

4: Log position *P* in detection log *L*

5: Lay high concentration of pheromones at *P*

6: **end if**

# 3 Results and Discussion

## 3.1 Pictographic overview of Developed Swarm Robotics Simulator for soybean crop row navigation

The Voronoi Zone Manager class divides the field into zones based on the robot's location. As illustrated in Fig. 2A, the field is partitioned into four distinct zones using the Voronoi tessellation (Du et al., 1999). This method enhances the overall efficiency and effectiveness of the swarm robotics system. In Fig. 2B, the NavMesh generated by the Unity AI engine is highlighted from the robot's camera perspective. Fig. 2C provides a top overview of the robots distributed across their assigned zones in a 40 m × 40 m simulated field. Typically, soybean farm sizes are very large in South Dakota, and simulating such quarter-section fields would require advanced GPUs with significantly higher than 24GB VRAM. However, a User Interface (UI) has been developed for such applications and is highlighted in Appendix 3. This UI facilitates interaction with the simulator, allowing for real-time adjustments and monitoring of the simulation parameters. When entering gameplay mode, the UI provides buttons for planting and controlling the robots, significantly reducing the VGPU memory load for a quarter-section field by optimizing hardware resource allocation.

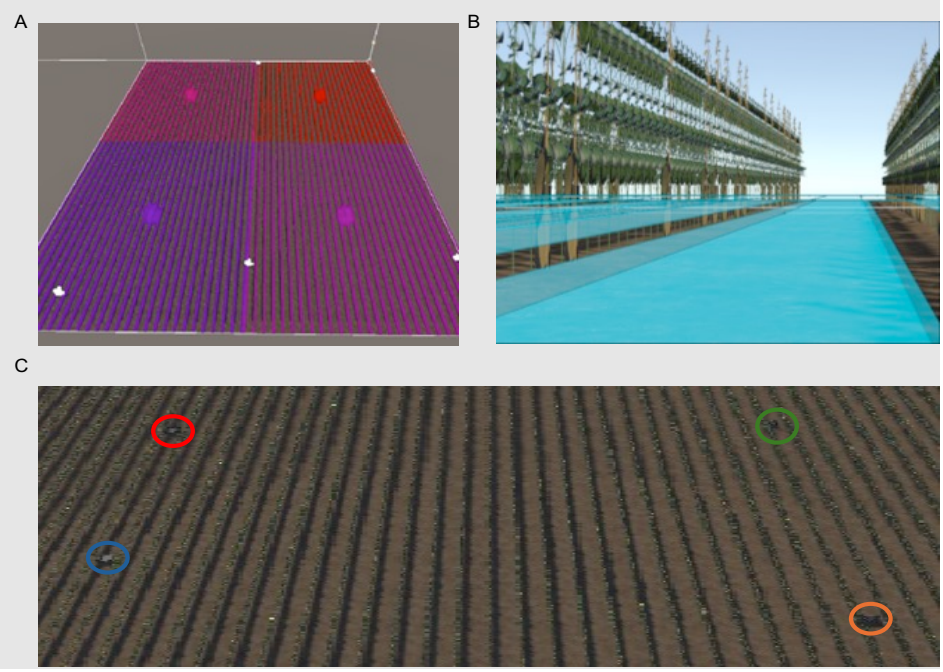


**Fig. 2 Simulation visualization after initiating V-ACO. (A) Four zones in the Voronoi diagram. (B) NavMesh seen from the robot's camera. (C) High-level overview of robots distributed in their zones. Robot 1 (Blue), Robot 2 (Orange), Robot 3 (Green), and Robot 4 (Red).**

### 3.2 Performance analysis of the V-ACO navigation framework for soybean row crop

The performance analysis of the V-ACO framework, as depicted in Fig. 3A, illustrates the spatial distribution of detected diseased plants by each robot, with the field partitioned into four distinct Voronoi zones. Each color represents a different robot, indicating their assigned zones and the detections made within those zones. This spatial distribution plot confirms that the Voronoi-based partitioning ensures comprehensive coverage of the field, with minimal overlap, thereby optimizing the efficiency of the swarm. Fig. 3B shows the coverage area achieved by each robot. The uniformity in coverage areas across the robots indicates that the V-ACO framework effectively ensures that each robot monitors its designated zone thoroughly. This is crucial for maintaining comprehensive field surveillance and minimizing gaps in coverage.

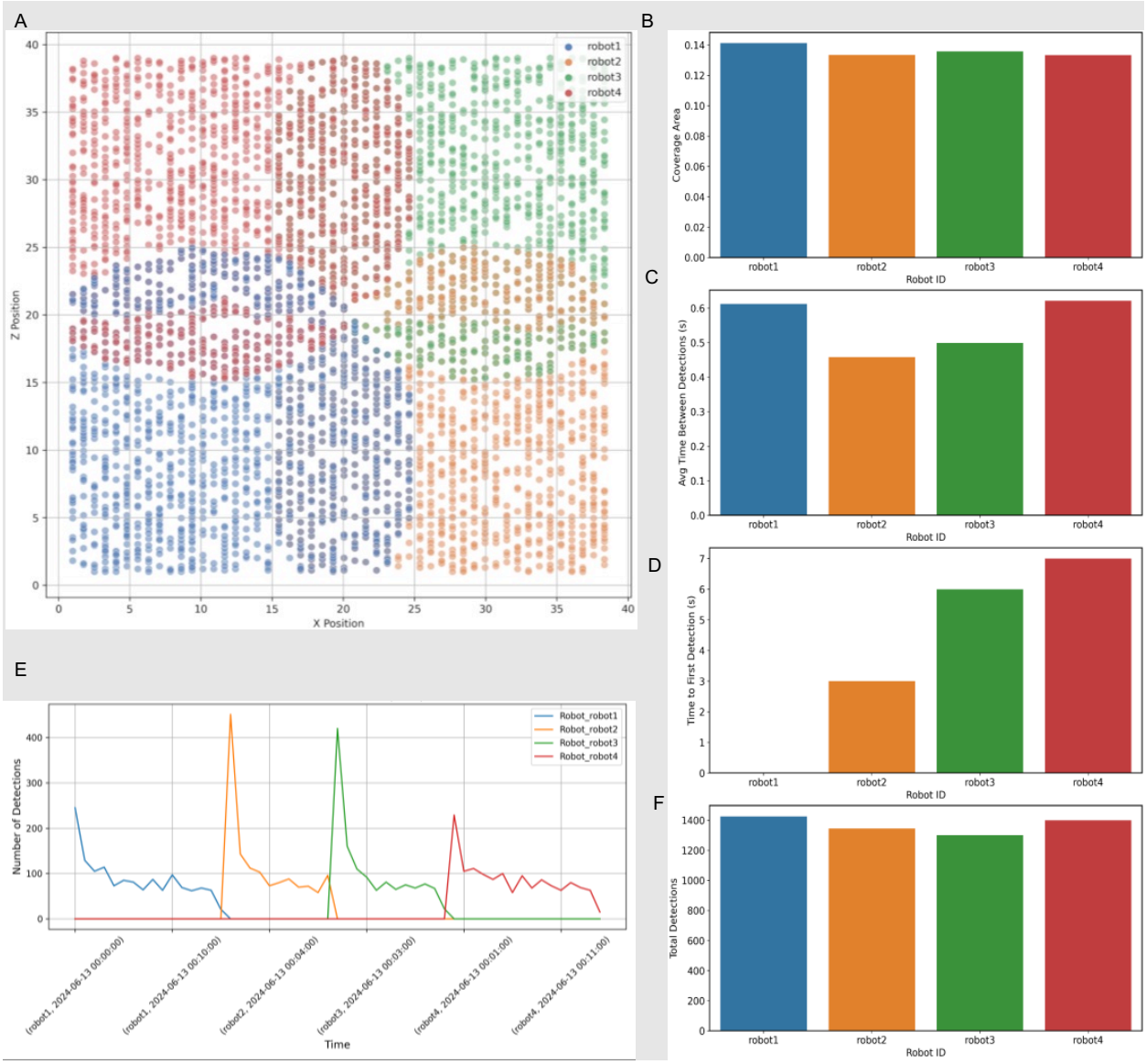


**Fig.3 Comprehensive performance metrics of the V-ACO framework for Soybean crop row navigation. (A) Spatial distribution of detected diseased plants by robots in the soybean field within Voronoi zones. (B) Coverage area achieved by each robot. (C) Average time between detections for each robot. (D) Time to first detection for each robot. (E) Temporal detection trends over time for each robot. (F) Total detections made by each robot.**

Fig. 3C presents the average time between detections for each robot. Robots 2 (Orange) and 3 (Green) show shorter average times between detections, indicating higher efficiency in identifying diseased plants compared to Robots 1 (Blue) and 4 (Red). This variation highlights potential areas for optimizing detection algorithms to ensure uniform performance across all robots. Fig. 3D displays the time taken for each robot to make the first detection of a diseased plant, based on a randomly distributed target plant in a simulated soybean field (20%). Robot 1 (Blue) shows a time

of 0 seconds for its first detection, indicating immediate detection upon starting. Robot 2 (Orange) is the second quickest among the robots, followed by Robot 3 (Green) and Robot 4 (Red), which took longer to detect their first diseased plant.

Fig. 3E tracks the temporal detection trends over time for each robot. This trend analysis is vital for understanding the temporal disease spread across the field and the robots' detection patterns. The graph shows fluctuations in detection counts over time, with peaks indicating periods of high detection activity. If one robot shows a high number of detections while others are low, it could imply that the robot is operating in a zone with a higher concentration of diseased plants, or that the detection algorithm for that robot is more efficient. This suggests the need for balancing detection algorithms and ensuring uniform coverage across the field to optimize overall detection efficiency. It's important to note that in this simulation, each robot is not limited to detecting specific types of diseases but rather detects any diseased plant within its operational zone. Fig. 3F summarizes the total detections made by each robot. All robots have made a substantial number of detections, with Robot 3 (Green) slightly ahead of the others. This indicates effective deployment, ensuring that all robots contribute significantly to the detection process. The slightly higher detection count for Robot 3 might suggest either a more efficient algorithm, a denser concentration of diseased plants in its assigned area, or optimal navigation and coverage strategies. These insights can guide further optimization of deployment strategies and algorithm tuning to enhance overall detection performance.

## 4 Conclusion

The performance analysis of the V-ACO framework demonstrates its potential in swarm robotics navigation and target identification in Unity 3D simulation. V-ACO improves the effective spatial distribution of detections within Voronoi zones and highlights the temporal detection trends, essential for understanding disease spread patterns, confirming comprehensive field coverage with minimal overlap (Figs. 3A and 3E). Comparing V-ACO with other path planning algorithms like Voronoi-only, ACO-only, and PSO will reveal the areas for improvement for this hybrid approach. V-ACO's integration of Voronoi tessellation and ACO provides a balanced approach, but optimizing detection algorithms and ensuring uniform performance across all robots remains crucial. Integrating Computer Vision (CV) and Deep Learning (DL) models using Unity's ML-Agents enhances the V-ACO framework's capabilities. This integration enables edge deployment and real-time processing, improving disease detection accuracy and efficiency. Advanced CV and DL techniques, such as Generative Adversarial Imitation Learning (GAIL), can further refine the swarm's behavior and detection performance (Ho & Ermon, 2016).

Current satellite and drone technologies are insufficient for managing rapid disease spread due to cloud cover, highlighting the importance of advanced path planning and immediate intervention capabilities. Parallel simulations of multiple scenarios will be conducted to test various environmental conditions and population dynamics. Field trials using a swarm of robots will validate the framework's efficacy in real crop situations. These trials will provide valuable insights, guiding further optimizations of the V-ACO framework and ensuring its practical applicability and effectiveness in precision agriculture. In conclusion, the V-ACO framework shows an efficient swarm robotic navigation approach for soybean fields, addressing the complexities of real-world environments and enhancing crop protection strategies.

## Acknowledgement

This research was funded by the Hatch Project (3AH777) and Multi Hatch Project (3AR730) of USDA NIFA through the South Dakota Agricultural Experimental Station at South Dakota State University.

# References


Albiero, D., Pontin Garcia, A., Kiyoshi Umezu, C., & Leme de Paulo, R. (2022). Swarm robots in mechanized agricultural operations: A review about challenges for research. In Computers and Electronics in Agriculture (Vol. 193). Elsevier B.V.

Crop Protection Network. (2023). Estimates of crop yield losses due to diseases and invertebrate pests: an online tool. https://loss.cropprotectionnetwork.org/.

De Melo, M. S. P., da Silva Neto, J. G., Da Silva, P. J. L., Teixeira, J. M. X. N., & Teichrieb, V. (2019, October). Analysis and comparison of robotics 3d simulators. In 2019 21st Symposium on Virtual and Augmented Reality (SVR) (pp. 242-251). IEEE.

Dorigo, M., & Stutzle, T. (2004). Ant Colony Optimization. MIT Press, Cambridge, MA.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28-39.

Du, Q., Faber, V., & Gunzburger, M. (1999). Centroidal Voronoi tessellations: Applications and algorithms. SIAM Review, 41(4), 637-676.

Dias, P. G. F., Silva, M. C., Rocha Filho, G. P., Vargas, P. A., Cota, L. P., & Pessin, G. (2021). Swarm robotics: A perspective on the latest reviewed concepts and applications. Sensors, 21(6), 2062.

Hamann, H. (2018). Swarm Robotics: A Formal Approach. In Swarm Robotics: A Formal Approach. Springer International Publishing. https://doi.org/10.1007/978-3-319-74528-2

Hu, Y. (2012). Swarm intelligence. https://guava.physics.uiuc.edu/~nigel/courses/569/Essays_Fall2012/Files/Hu.pdf

Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. Advances in neural information processing systems, 29.

Karunarathna, D., Jaliyagoda, N., Jayalath, G., Alawatugoda, J., Ragel, R., & Nawinne, I. (2023). Mixed-Reality based Multi-Agent Robotics Framework for Artificial Swarm Intelligence Experiments. IEEE Access.

Lim, S., Wang, S., Lennox, B., & Arvin, F. (2021, February). Beeground-an open-source simulation platform for large-scale swarm robotics applications. In 2021 7th International Conference on Automation, Robotics and Applications (ICARA) (pp. 75-79). IEEE.

Lynxmotion. (n.d.). A4WD3 rugged rovers. Retrieved June 13, 2024, from https://www.lynxmotion.com/a4wd3-rugged-rovers/

Mueller, D. S., Dorrance, A. E., Derksen, R. C., Ozkan, E., Kurle, J. E., Grau, C. R., ... & Pedersen, W. L. (2002). Efficacy of fungicides on Sclerotinia sclerotiorum and their potential for control of Sclerotinia stem rot on soybean. Plant Disease, 86(1), 26-31.

Okabe, A., Boots, B., Sugihara, K., & Chiu, S. N. (2009). Spatial tessellations: concepts and applications of Voronoi diagrams.

Şahin, E. (2004, July). Swarm robotics: From sources of inspiration to domains of application. In International workshop on swarm robotics (pp. 10-20). Berlin, Heidelberg: Springer Berlin Heidelberg.

South Dakota State University Extension. (2020, August 31). iGrow soybeans: Best management practices for soybean production. Retrieved June 13, 2024, from https://extension.sdstate.edu/igrow-soybeans-best-management-practices-soybean-production

Unity Technologies. (2022). NavMesh. In Unity Script Reference. Retrieved June 13, 2024, from https://docs.unity3d.com/ScriptReference/AI.NavMesh.html

Unity Technologies. (n.d.). Unity User Manual. Retrieved from Unity Documentation https://docs.unity3d.com/Manual/index.html

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & Van Mulbregt, P. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods, 17(3), 261-272.

Wang, D., Zhang, B., Zhou, J., Xiong, Y., Liu, L., & Tan, Q. (2024). Three-dimensional mapping and immersive human–robot interfacing utilize Kinect-style depth cameras and virtual reality for agricultural mobile robots. Journal of Field Robotics.

XFrog. (n.d.). Library Agriculture. Retrieved from https://www.xfrog.com/product-page/library-agriculture

Xiong, C., Chen, D., Lu, D., Zeng, Z., & Lian, L. (2019). Path planning of multiple autonomous marine vehicles for adaptive sampling using Voronoi-based ant colony optimization. Robotics and Autonomous Systems, 115, 90-103.

Yang, X. S. (2020). Nature-inspired optimization algorithms. Academic Press.

Zhou, X., Wen, X., Wang, Z., Gao, Y., Li, H., Wang, Q., Yang, T., Lu, H., Cao, Y., Xu, C., & Gao, F. (2022). Swarm of micro flying robots in the wild. In Sci. Robot (Vol. 7).

# Appendices

## Appendix 1 Soybean simulation plant prefab models from Xforge agriculture library

In our project, we utilized a 3D model of soybean plants to simulate the field environment (XFrog, n.d.). This model can be found on TurboSquid's website. The soybean section from the agriculture library in XFrog contains nine variations depicting various growth stages. For the 3D simulation, variations 7 and 9, which are 94 cm and 100 cm, respectively were chosen (Fig. S1). The variation 9 resembles a dried plant or overly mature plant, which is used to identify diseases for simulation purposes.

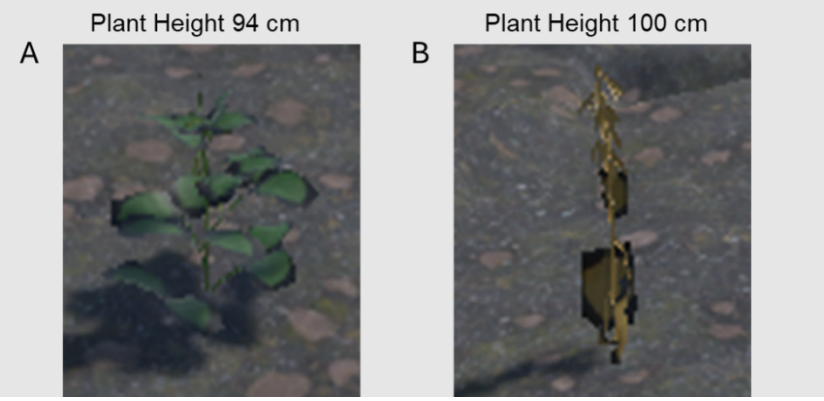


**Fig. S1 Soybean simulation plant prefab models from the Xforge agriculture library. (A) Healthy plant. (B) Diseased plant (XFrog, n.d.).**

## Appendix 2 Lynxmotion AWD4 Rugged abstract model and Nav Mesh Agent parameters used in simulation study

The A4WD3 rover, selected for this simulation research, is particularly suited for agricultural research applications due to its ability to navigate up to 25-inch crop spacing in crops. Measuring 451.5 mm (17.776 inches) in width, the rover navigates effectively between rows, minimizing potential damage to plants (Fig. S2). Its robust construction, featuring a durable machined aluminum frame and G10 composite plates, ensures resilience against environmental elements like water and dust. Additionally, the propulsion system, consisting of four 12 V DC motors with 27:1 metal planetary gears and rear encoders, along with four sets of sprockets and reinforced continuous rubber tracks, provides precise navigation and exceptional traction across various agricultural terrains, including uneven fields (Lynxmotion, n.d.).

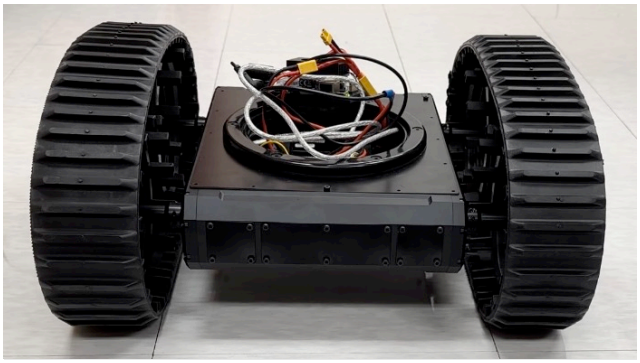

**Fig. S2. The robot model used in the simulation is based on the Lynxmotion A4WD3 rugged rover platform.**

**Table S1. Nav Mesh Agent Parameters Used in Simulation Study**

| Parameter | Value in Unity | Real World Measurement (meters) | Real World Measurement (feet/inches) |
|---|---|---|---|
| **Agent Type** | Humanoid | - | - |
| **Base Offset** | 136 | 136 meters | 446.194 feet |
| **Speed** | 1 | 1 meter/second | 3.281 feet/second |
| **Angular Speed** | 90 | 90 degrees/second | 90 degrees/second |
| **Acceleration** | 5 | 5 meters/second² | 16.405 feet/second² |
| **Stopping Distance** | 0.1 | 0.1 meters | 3.937 inches |
| **Auto Braking** | Enabled | - | - |
| **Obstacle Avoidance Radius** | 20 | 20 meters | 65.617 feet |
| **Height** | 0.5 | 0.5 meters | 19.685 inches |
| **Quality** | High Quality | - | - |
| **Priority** | 50 | - | - |
| **Auto Traverse Off Mesh Link** | Enabled | - | - |
| **Auto Repath** | Enabled | - | - |
| **Area Mask** | Everything | - | - |
| **Agent Radius** | 0.4572 | 0.4572 meters | 18 inches |
| **Agent Height** | 0.1 | 0.1 meters | 3.937 inches |
| **Max Slope** | 44.8 degrees | 44.8 degrees | 44.8 degrees |
| **Step Height** | 0.02 | 0.02 meters | 0.787 inches |
| **Drop Height** | 0 | 0 meters | 0 feet/inches |
| **Jump Distance** | 0 | 0 meters | 0 feet/inches |

## Appendix 3 Unity UI for planting quarter section soybean field

Algorithm S.1 instantiates soybean plants in a quarter-section field within Unity. It uses two prefabs, one for diseased plants and one for healthy plants, and places them in rows and columns with specified spacing and margins. The selection of plant type is randomized with a 20% probability for diseased plants. Once the user clicks play and enters Game mode, using these snippets mentioned in Fig. S3 in the Unity interface helps efficiently load and manage a large scene such as a quarter-section field.

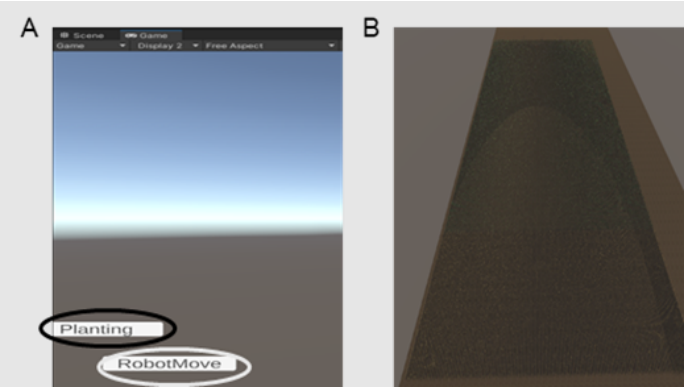


**Fig. S3. Unity User Interface (UI) for planting and robot movement in a quarter-section soybean field simulation. (A) The panel shows the UI buttons for planting and robot movement. (B) The right panel displays the planted quarter section soybean field.**

**Algorithm S.1** Plant Soybeans in a Quarter-Section Field

**1: Input:**

2: *prefab20*: Prefab for diseased soybean plant

3: *prefab80*: Prefab for healthy soybean plant

4: *quarterSectionFieldWidth* ← 1012.462 meters

5: *quarterSectionFieldLength* ← 640.0 meters

6: *rowSpacing* ← 0.762 meters

7: *plantSpacing* ← 0.1016 meters

8: *margin* ← 0.6096 meters

9: **Output**:

10: A field of soybean plants instantiated in Unity

11: Create a new GameObject named *SoybeanQuarterSectionField*

12: **for** each position *x* from margin to *quarterSectionFieldWidth – margin* incrementing by *rowSpacing* **do**

13: **for** each position *z* from margin to quarterSectionFieldLength – *margin* incrementing by *plantSpacing* do

14: Randomly select a prefab (*prefab20* or *prefab80* ) with a 20% probability for *prefab20*

15: Create a new plant instance at position (*x, 0, z*)

16: Set the new plant instance's parent to *SoybeanQuarterSectionField*

17: **end for**

18: **end for**