2D Swarm Construction

AE4350: Bio-Inspired Intelligence and Learning for Aerospace Applicatons

Richard Adam (6149022)

Delft University of Technology

TU Delft

Introduction

The field of swarm robotics has emerged as a heavily-investigated area of study within the broader domain of multi-agent systems. Inspired by the self-organizing behaviors observed in biological groups such as ant colonies, bee swarms, and bird flocks, the key idea in swarm robotics is to have each agent follow relatively simple rules, but create complex and consistent emergent behaviors when there are many swarm agents interacting. Unlike traditional robotic systems that often rely on a centralized piece of expensive hardware designed to have very complex behaviors, swarm robotics systems typically consist of many simple and low-cost hardware components. The field attempts to create emergent behaviors where global coordination from a swarm rivals the coordination of centralized robots in their particular fields. If globally emergent behaviors can rival centralized robotic systems, swarms pose the benefit of enabling higher adaptability, fault tolerance, and scalability.

One of the most promising applications of swarm robotics is construction, as multiple agents can work together to build physical structures in a decentralized manner. There are significant challenges to swarm robotic construction, though, as agents must construct global structures without having any pre-defined map of the structure itself. There are some proposed solutions to this, like using stigmergy, or marking of the environment to define certain behaviors. Another solution, the one this design implements, is to have swarms build quite simple structures that do not require a map, as even simple hexagonal structures can be incredibly useful in aerospace structural design. For example, the honeycomb lattice structure is incredibly common in Aerospace engineering due to its high strength to weight ratio.

The algorithm explored in this assignment, inspired by a recent paper from the University of Pennsylvania [1] attempts to investigate the construction of honeycomb lattices by a swarm of simple agents. A comprehensive analysis of the algorithm is conducted, detailing its structure, behavior, and how specific parameters of the model change the results of the system.

Algorithm Overview

The algorithm revolves around a far-future in-space construction scenario where there is some centralized raw building material, for example a large piece of metal sent from Earth or a metallic asteroid already in space. The agents are deployed centrally at the location this building material, and they can gather chunks of the material for construction. They melt the material down and extrude it to create straight-line struts, or links in the hexagonal lattice structure, somewhat like 3D printers do. The agents have the ability to follow the struts, and when reaching a junction of struts they choose randomly which path to follow. After extruding their new strut into the structure, they must return to the centralized material source to gather more building material, which they can do via following an rf signal, which is placed at the material source, up-gradient. They must gather for some pre-defined time before going back to follow the structure and deposit more struts.

The agential behavior for the algorithm is defined in the agents.py script, which defines how each agent operates within the 2D environment. The script sets the behavioral rules that govern agent's decisions when they are searching for, following, and depositing a strut, as well as returning and gathering material. Each swarm agent operates as a basic state machine with six possible states the system can be in: searching, moving to a strut, following a strut, depositing, homing, or gathering. The behavior in each state is defined as follows:

1. Searching: Random walk search that is biased toward the center of the construction region if the agent is far from the center. This bias helps accelerate the early stages of the construction. If an agent detects a strut in its vision area, it will set that strut as a target and switch to the ”Moving To Strut” state.
2. Moving to Strut: Movement direction is set toward the target strut and remains constant. When within a threshold distance, attach to the strut and change state to ”Following Strut”.
3. Following Strut: Choose randomly which direction to follow when it first transitions from ”Moving to Strut”. When reaching the end of a strut, if there are two attached struts, randomly chose one of them to follow. If there is only one attached strut, turn 120deg away from both struts and switch to ”Depositing” state. If there are no attached struts, choose randomly between turning +60deg

or -60deg from the current heading and switch to ”Depositing” state.

4. Depositing: Deposit strut until it is the target length. Optionally, one can specify a standard deviation to apply some imperfections to the length deposited. When reaching the end of the deposition target length, check if there are any struts within the body size and, if there is, attach the end of the generated strut to the start of the detected strut. When depositing is finished, switch to ”Homing” state.
5. Homing: Detach from the strut and follow the rf signal up-gradient back to the gathering area. Once the gathering area has been reached, switch to ”Gathering” state.
6. Gathering: Gather materials for a pre-determined amount of time. Once materials are gathered, switch to the ”Searching” state.

In all states, to avoid collisions, the agents also avoid any occupied strut.

Simulation Setup

In order to fit the decentralized algorithm into a centralized computer simulation, there are some implications on the algorithm that should be noted. One of these is the fact that the simulation uses a list of struts, endpoints, and connections. In the simulation, agents check against this centralized map to tell when a strut is in range or how many connections an endpoint has. This could easily be replaced by cameras or ranging sensors in an actual swarm robotic system, but this centralized approach must be used for the simulation. The agents, as state machines, also have some form of memory in that they can move in a particular direction or maintain a particular state, but their decision-making remains entirely reactive.

In order to test the agent logic, a high-level behavioral simulation was set up and tested for multiple different parameters. For each run of the simulation, one can specify:

1. Number of Agents
2. Amount of Timesteps
3. Target Strut Length
4. Maximum Strut Error
5. Speed of Deposition

tests were conducted on seventeen different scenarios in order to view the sensitivity of the system with respect to these parameters. The seventeen scenarios are as follows:

Scenario:	Agents:	Timesteps [ms]	Target Len. [mm]	Err. Bounds [mm]	Dep. Speed [mm/ms]
1	1	50000	4	0	0.1
2	3	50000	4	0	0.1
3	5	50000	4	0	0.1
4	7	50000	4	0	0.1
5	10	50000	4	0	0.1
6	20	50000	4	0	0.1
7	5	50000	1	0	0.1
8	5	50000	7	0	0.1
9	5	50000	10	0	0.1
10	5	50000	4	0.5	0.1
11	5	50000	4	1	0.1
12	5	50000	4	2	0.1
13	5	50000	4	0	0.4
14	5	50000	4	0	1
15	5	5000	4	0	0.1
16	5	10000	4	0	0.1
17	5	25000	4	0	0.1

Results

Stability over Constant Conditions

Since the behavior of the agents is random, it is worth a statistical analysis of how stable the results are across a single initial condition. The condition chosen is scenario 3, where there are 5 agents, each with a target length of 4 mm and no errors in deposition. The deposition speed is 0.1 m/s and the trial takes place over 50000 ms. Ten trials were simulated in this condition, and the total number of generated struts for the ten trials are:

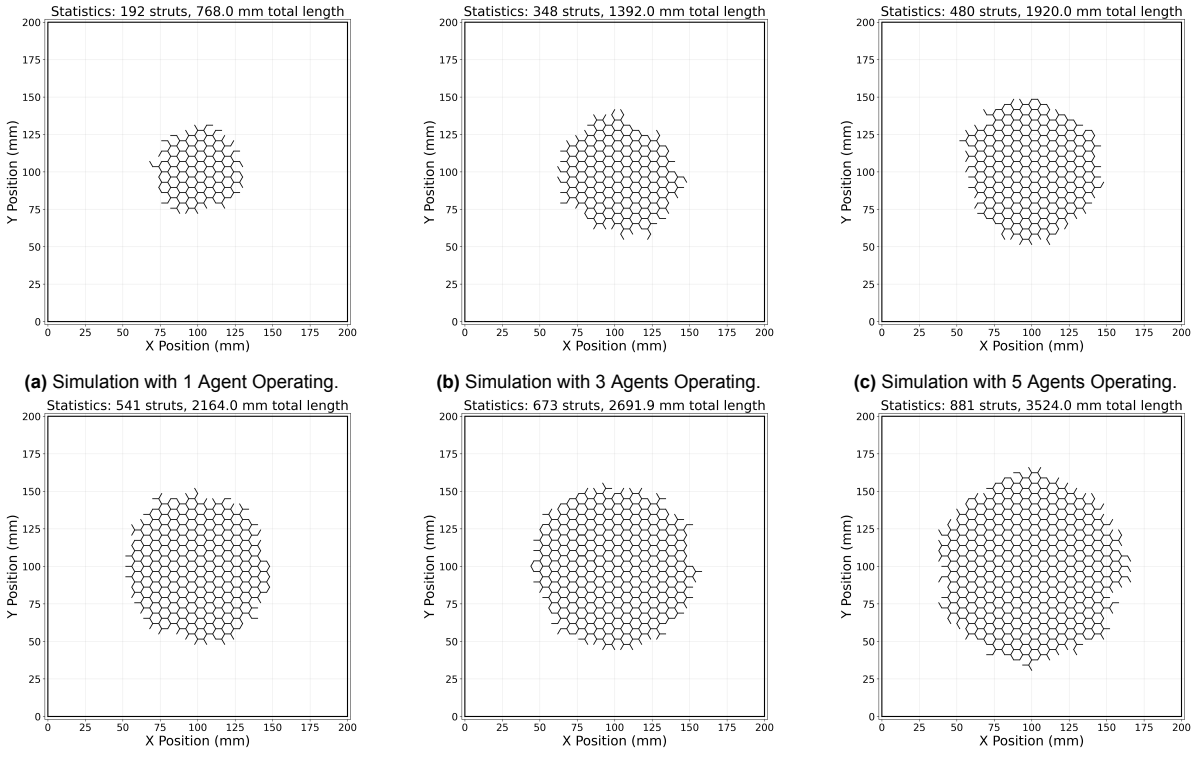
Run	1	2	3	4	5	6	7	8	9	10
Struts	480	482	469	479	469	470	477	463	489	475

Table 1: Total number of struts generated for the ten trials in Scenario 3.

Of these runs, the most struts generated was 489 and the least generated was 463, leaving a range of 26 struts. The standard deviation of the number of struts created is 7.3 struts. This is about 1.5% of the mean number of struts created, 475.3, so the number of struts created remains relatively consistent across runs despite some outliers.

Sensitivity to Agent Number

Next, it is useful to understand how sensitive the system is to the number of active agents. Simulations were run for 1, 3, 7, 10, and 20 active agents, and the results can be seen in Figure 1:



(a) Simulation with 1 Agent Operating.
(b) Simulation with 3 Agents Operating.
(c) Simulation with 5 Agents Operating.
(d) Simulation with 7 Agents Operating.
(e) Simulation with 10 Agents Operating.
(f) Simulation with 20 Agents Operating.

Figure 1: Subplots with Varying Number of Agents

To better understand these results, the total length of struts and radius of the circle that circumscribes the structure is compared for all of the step sizes. The results can be seen in Figure 2:

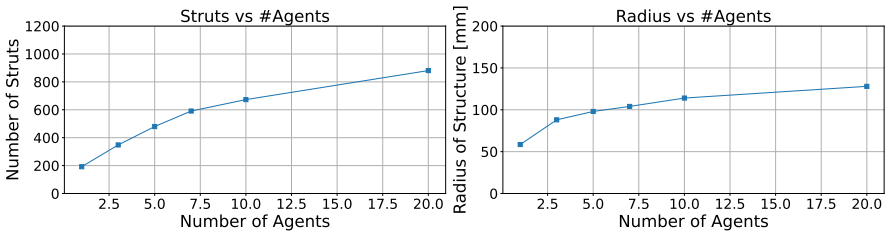
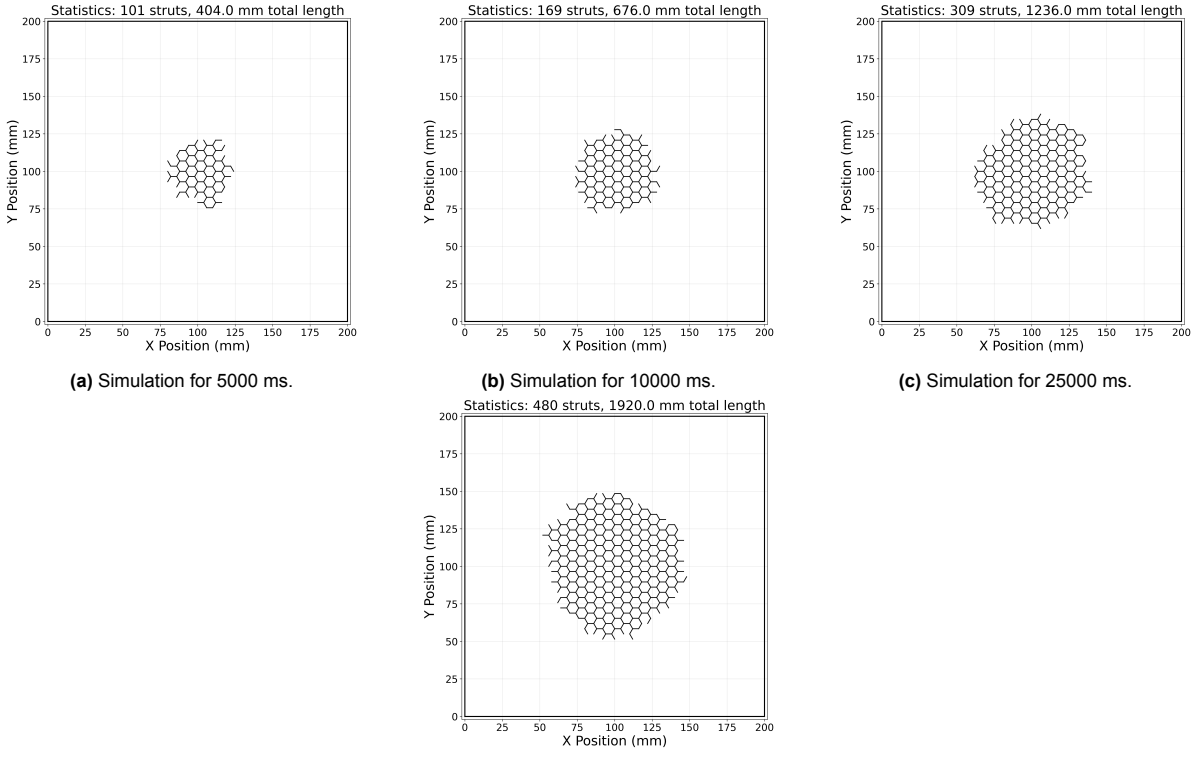


Figure 2: Overall comparison of struts created and structure radius as a function of the number of active agents.

Clearly, increasing the number of agents has diminishing returns on both strut length and radius of created structure. Based on the plots, the struts and radius scale as $\mathcal{O}(\log(n))$ where n is the number of agents, suggesting that for this particular algorithm, there is a sweet spot where the number of struts is the cheapest relative to the total number of agents necessary to build them.

Sensitivity to Number of Steps

Along with the number of agents, the total number of steps is an important parameter to examine. The results for 5000 ms, 10000 ms, 25000 ms, and 50000 ms can be seen for the 5-agent case in Figure 3:



(a) Simulation for 5000 ms.

(b) Simulation for 10000 ms.

(c) Simulation for 25000 ms.

(d) Simulation for 50000 ms.

Figure 3: Subplots with Varying Step Size

Clearly, like the agent number, with the current algorithm there are diminishing returns to strut generation as step size increases. This can be seen in the plots of strut generation and structure radius in Figure 4:

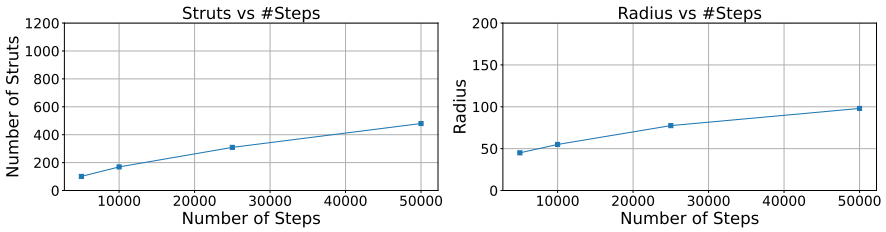
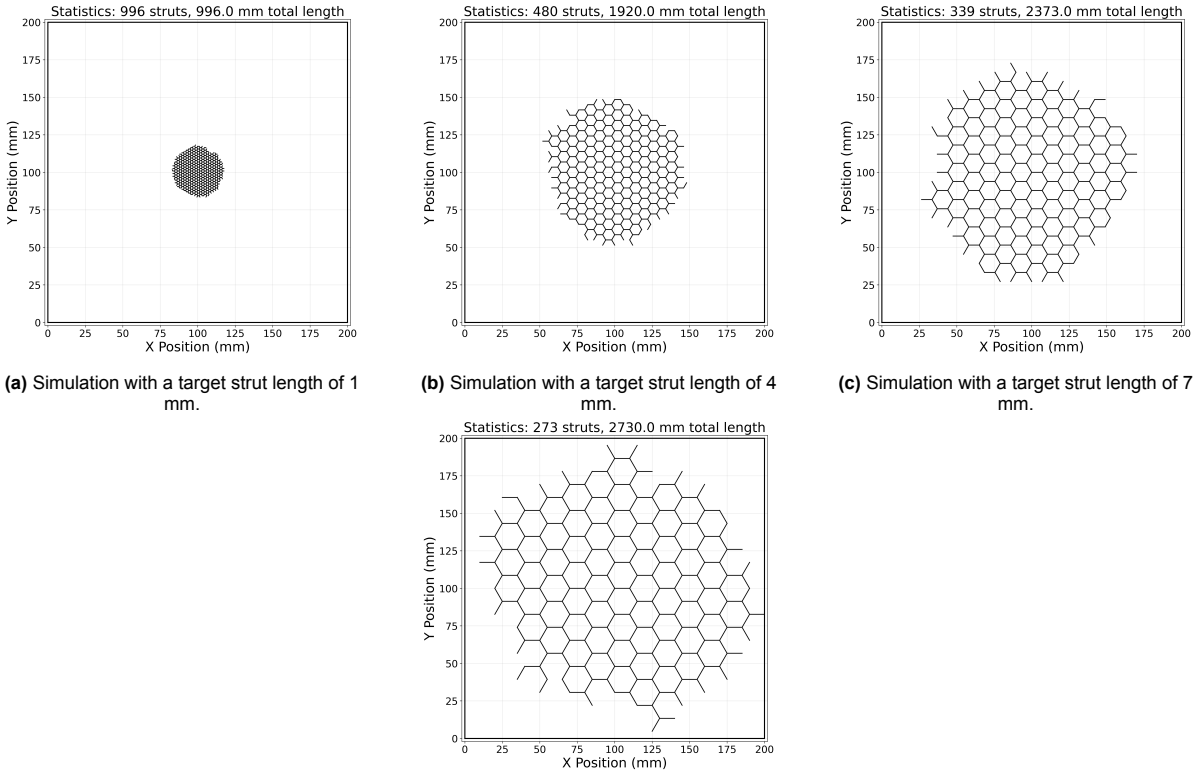


Figure 4: Overall comparison of struts created and structure radius as a function of the number of total simulation steps.

Over the timescales examined, the system seems to be just slightly sublinear in its growth of struts created and structural radius with respect to the total time elapsed.

Sensitivity to Target Length

Another important parameter for the structural properties of the lattices is the length of the struts themselves. Strut lengths of 1 mm, 4 mm, 7 mm, and 10 mm were simulated and can be seen in Figure 5:



(a) Simulation with a target strut length of 1 mm.

(b) Simulation with a target strut length of 4 mm.

(c) Simulation with a target strut length of 7 mm.

(d) Simulation with a target strut length of 10 mm(1 cm).

Figure 5: Subplots with Varying Target Lengths

The resulting plot of struts generated and structure radius as a function of target strut length can be seen in Figure 6:

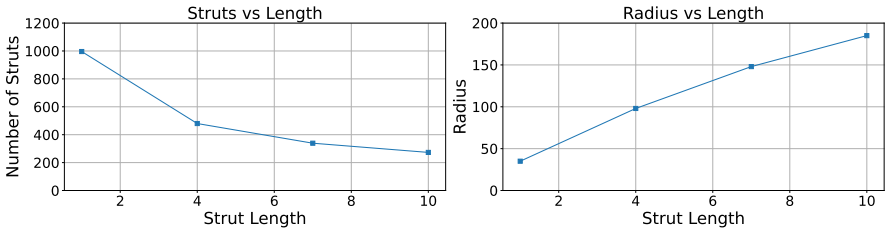
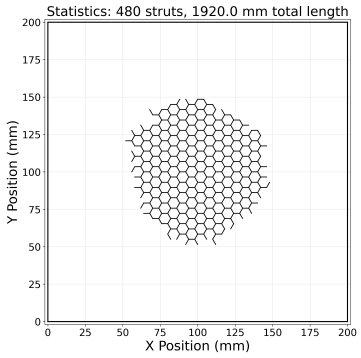


Figure 6: Overall comparison of struts created and structure radius as a function of the target strut length.

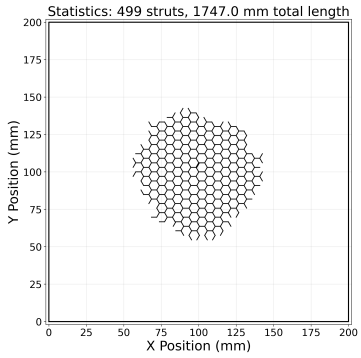
Logically, the smaller the target strut size the more total struts are created. On the other hand, the total radius of the structure increases almost linearly with the target strut length.

Sensitivity to Length Errors

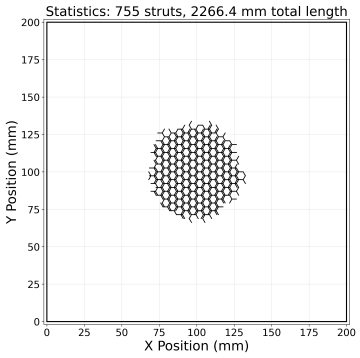
In real-life extrusion systems, there are small errors in the total length extruded. Perfectly hexagonal structures assume perfect strut lengths, so errors in strut lengths can make large differences in the overall structure. To investigate this, error bounds of 0 mm, 0.5 mm, 1 mm, and 2 mm were simulated for a target strut length of 4 mm. The results can be seen in Figure 7:



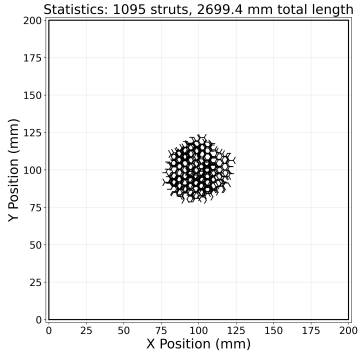
(a) Simulation with no error in length of struts.



(b) Simulation with ± 0.5 mm error in length of struts.



(c) Simulation with ± 1 mm error in length of struts.



(d) Simulation with ± 2 mm error in length of struts.

Figure 7: Subplots with Varying Length Errors

The resulting plots of total struts and structure radius as a function of strut error limits can be seen in Figure 8:

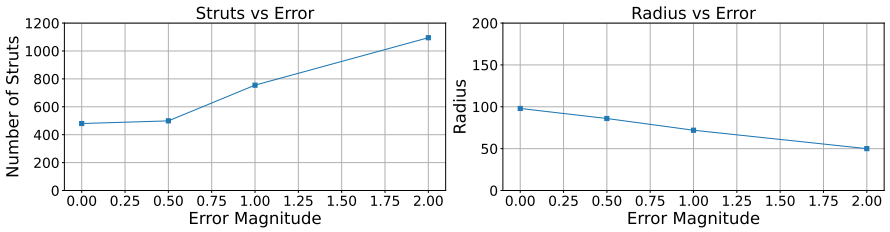
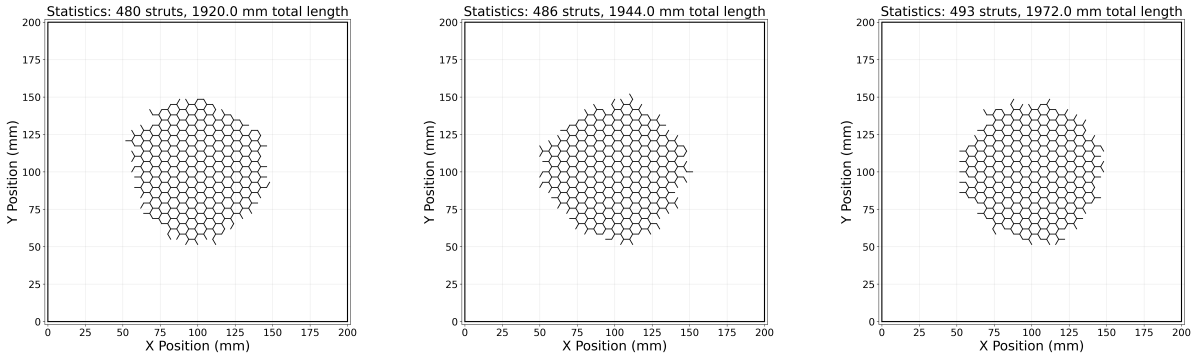


Figure 8: Overall comparison of struts created and structure radius as a function of the error bounds for strut length.

As the error increases, the total number of struts goes up, but the total radius of the structure goes down. This means not only is the spatial efficiency of the material deposited going down, but the honeycomb structure completely deteriorates into either a double honeycomb structure as Figure 7c shows, or a messy jumble of struts as Figure 7d shows.

Sensitivity to Building Speed

The final parameter examined is the speed at which the agents extrude material. The speeds of 0.1 mm/s, 0.4 mm/s, and 1 cm/s were examined, and the results can be seen in Figure 9:



(a) Simulation with a deposition speed of 0.1 mm/s.

(b) Simulation with a deposition speed of 0.4 mm/s.

(c) Simulation with a deposition speed of 1 cm/s.

Figure 9: Subplots with Varying Deposition Speed

The plots looks quite similar, and this can be confirmed by looking at the plot of struts created and radius as a function of deposition speed in Figure 10:

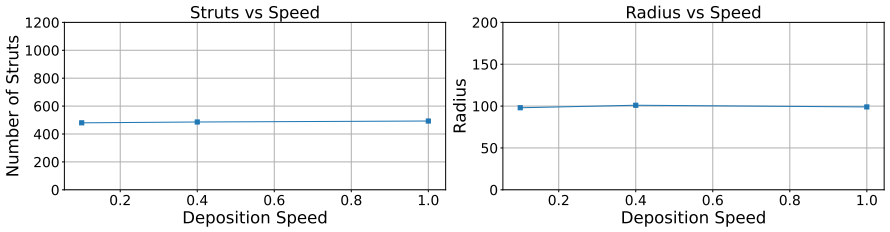


Figure 10: Overall comparison of struts created and structure radius as a function of the strut deposition speed.

The deposition speed has very little effect on the total growth of the structure at the timescale of 50000 ms, showing up as an almost horizontal line on the plot of struts generated and radius of structure.

Discussion

Almost all of these findings can be explained by a simple characteristic of the algorithm, one that must be improved before implementing the swarm algorithm for large-scale hexagonal lattice production. In all cases, as the structure generates more and more struts, the rate at which struts are created goes down. In both the cases where there are more agents and the cases where there is more construction time, the system grows rapidly early in the process, but begins to plateau. This is because of the behavior in the ”Following Strut” state, where an agent reaches the end of a fully connected strut and chooses randomly between the two attached struts. This means that there is no mechanism that is pushing the agents to leave the center of the structure other than chance, so as the structure gets bigger, the random behavior means that the agents will generally be further from the perimeter where they can deposit, so the rate of structural generation goes down. There is a potential solutions to this. This algorithm already assumes that there is some signal source that can be followed up-gradient to find the gathering zone. That same signal could theoretically be used to bias agents toward choosing struts that get them down-gradient, or farther from the signal source, which would accelerate the building rate as the structure grows since agents would have a statistical bias toward the perimeter of the structure.

There are also other interesting findings in the results, starting with the sensitivity to length errors. Extrusion-based systems are somewhat error-prone, so the fact that relative errors can cause significant reductions in spatial efficiency as Figure 7 shows is a big problem for cheaper extrusion-based solutions. If the system were implemented for in-space construction using in-space materials, it may be worth using a heterogeneous multi-robot system that has one component creating high-precision struts and a swarm of smaller cheaper robots constructing the structure from pre-made struts. It is worth noting, though, that the behavior at the end of the ”Depositing” state where the robot ”snaps” the end of the strut to struts within a certain range is useful to mitigate these imperfections, as Figure 7b shows, having a very low impact of errors on strut creation and structure radius. In addition, if there is a trade between strut deposition speed and strut length accuracy, based on the findings of Figures 9 and 10 it seems that the accuracy should be favored, as deposition speed does not significantly increase strut generation or structural radius at higher time-scales.

Finally, the decision for length of struts is an important one for the particular structure being designed. Figure 5 shows that using higher target strut length can increase the overall radius significantly and make the structure more space-efficient, but that may cost the structure some stiffness. In all likelihood, the choice will depend on the structure created, as something like a stationary space station may not have strict vibrational requirements, but some part of a rocket-powered interplanetary vehicle may have extremely strict vibrational requirements. In addition, it is worth noting that one of the reasons smaller target strut lengths lead to such a reduced radius is that agents following random struts stay in the center more often as described earlier, so the down-gradient bias adjustment to the algorithm could help remedy this issue.

Conclusion

The proposed method of hexagonal lattice construction using a swarm of simple state-machine agents poses an interesting application for in-space construction. The simple six-state agent behavior ensures that there is low computational cost for each agent and low hardware complexity for agent design. Problematically, though, the findings show that the current algorithm has diminishing returns when increasing both agent number and construction time. This is problematic because it nullifies one of the attractive attributes of swarm systems: scalability. This issue can potentially be mitigated by implementing a down-gradient bias to the random decisions for the ”Following Strut” state for each agent. There are also important implications on the findings for strut errors. If strut errors are higher than the range of ”snapping” at the end of the Deposing state, the structure of the lattice deteriorates quickly, and the spatial efficiency of strut placements decreases as well, using more material for the same amount of area. The finding that deposition speed does not significantly increase the struts generated suggests that lattice accuracy should be favored over the speed of each agent. Finally, target strut length is an important factor in the system, with an almost linear relationship between target strut length and structural radius suggesting that for scenarios without high stiffness and vibrational stability requirements, higher target lengths can be incredibly useful. Overall, the findings show that swarm-based hexagonal lattice construction is certainly worth future investigation for in-space construction.

References

[1] Liu, J., Zhu, X., Gosrich, W., Yim, M., & Raney, J. R. (2025). Design of nondeterministic architected structures via bioinspired distributed agents. Science Advances, 11(20). https://doi.org/10.1126/sciadv.adu8260

Link to Codebase

`https://github.com/radam1/AE4350_Hexagonal_Swarm_Construction`