JANUARY 31, 2025

A MODULAR SELF ASSEMBLING ROBOTIC SYSTEM

Maxime Buck

TABLE OF CONTENTS

1. Introduction

1.1 Short Project Summary

My project involves the conceptualization and development of a modular self-assembling robotic systems. Through its ability to form any complex configuration, the system is highly adaptable to various scenarios and environments.

1.2 Background

After watching "Big Hero 6", I felt amazed by the applications of the so called “microbots”. From then on, I wondered what was possible and my research led me to modular robotics.

Main character Hiro showcasing the «Microbots»

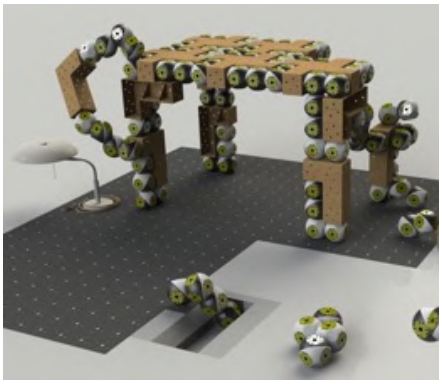
1.3 What Are Modular Robots?

Self-Assembling modular robotic systems are a type of swarm robotics where individual modules assemble into various configurations to accomplish tasks.

- Each module has limited capabilities (degrees of freedom, mobility), but when assembled, they create a system with extensive adaptability in any environments, where normal robots are constructed for only a certain task.
- Replacement of failed modules is simple, with a new module taking the place of the failed one in the configuration.
- Automatically self-assembling eliminates the need for prolonged human interaction and enables the formation of complex structures.

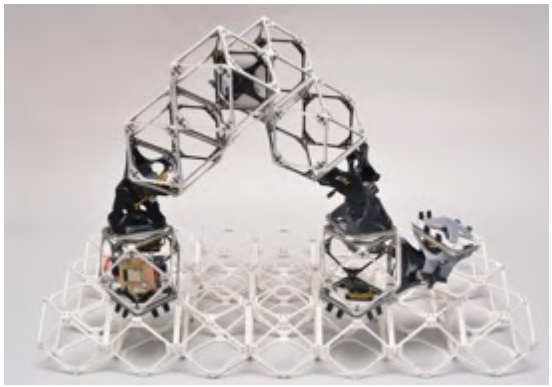
Use cases of modular robotic systems:

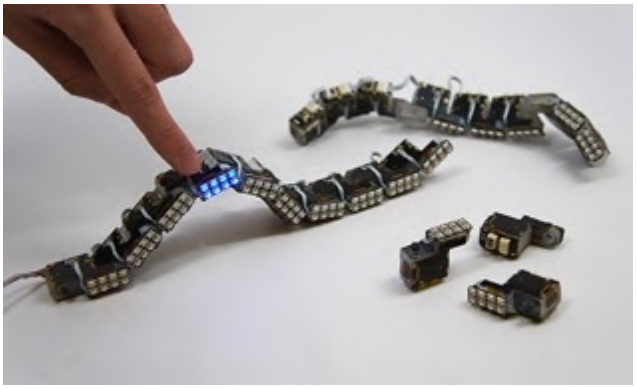
- Capable of carrying objects and equipped with special modules for performing all kinds of actions on external objects.
- Can be controlled to access dangerous environments (e.g., extreme temperatures, collapsed buildings, radioactive areas).
- Diagnose problems and reorganize themselves to provide solutions to the tasks they face.
- Adaptable furniture design, allowing for dynamic changes in furniture configuration based on user needs.



1.4 Motivation

Current robots are built for specific tasks in which they excel but lack adaptability for any other tasks. They cannot adjust to changes, rendering them unsuitable for unknown environments.





While modular systems offer a promising solution, there are three main problems with most current projects:

- Most are limited in tasks and remain bulky despite efforts to minimize size because each module is designed as an individual robot capable of computing, localizing, moving, and housing a battery, resulting in bulkiness.
- Single modules lack independent movement due to their limited degree of freedom. This limitation restricts assembly possibilities and the ability to adapt to diverse environments. Also, if a robot disconnects or gets lost, it will be unable to get back to the colony.
- Modular robot projects fail to realize their full potential due to the inability to show promising simulations. Often only trying to use a tiny number of real bots which can't show their true potential leading to a possible lack of funding.

2. Project description

2.1 Idea & Brainstorming

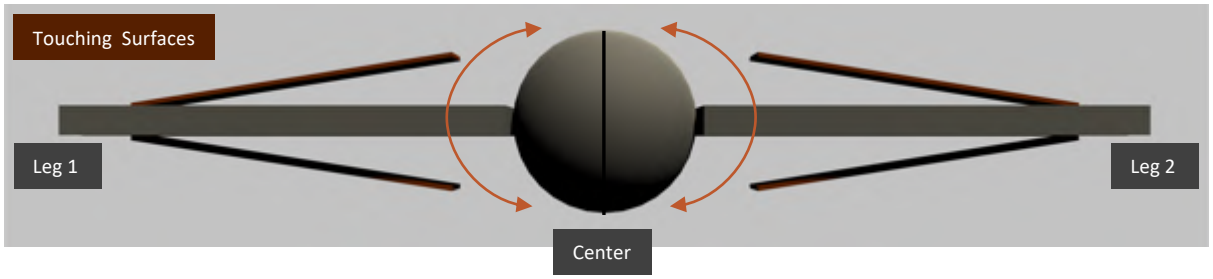
To address the challenges outlined earlier, I propose several solutions:

I aim to reduce each module's size by eliminating bulky components like large batteries, multiple sensors, and excessive computing. Inspired by bee swarms and other insect colonies, I envision a system with a central coordinator and numerous worker modules, focusing on developing the worker module as the system's backbone.

The goal is to create a self-assembling modular robot system with modules that have high degrees of freedom, efficient mobility across various surfaces, and autonomous movement.

Because of the project's complexity and my primary interest in programming, I will first explore the construction of a single module to assess the project's feasibility. This approach allows me to concentrate on key aspects such as communication, simulation, and the organizational structure of the system.

2.2 Key concept



My key concept shown in this picture shows a worker module characterized by its central core housing all components, along with two legs. Each leg possesses the capability to rotate within a two-dimensional plane. Additionally, the central core is divided into two halves, allowing rotation around a central axis.

2.3 Requirements of a Worker Module

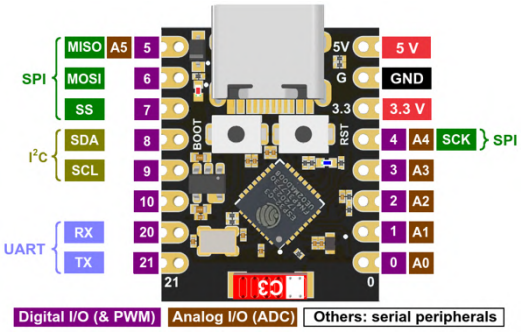
The size of a single module plays a crucial role in determining the complexity and potential of a larger structure. With this in mind, module size becomes a key factor in the next steps of development. That's why, throughout the entire design phase, the primary focus was on size management and finding the smallest components available.

Stable Attachment

Integrating small neodymium permanent magnets with electromagnets enables the creation of a custom energize-to-release mechanism. This design ensures that modules maintain a strong connection without the need for continuous current. Turning the electromagnet on at any moment neutralizes the magnetic field of the magnet releasing the modules from each other.

Receiving and Executing Commands

Each bot will be equipped with an ESP32-C3 microcontroller. I switched to the ESP32-C3 because it is significantly cheaper, very small, and offers more GPIO pins than the Seeeduino. Additionally, it has the same tiny size while providing Bluetooth capability. Finally, it can be programmed via USB-C and offers a total of 13 GPIO ports, providing more flexibility than the Seeeduino.

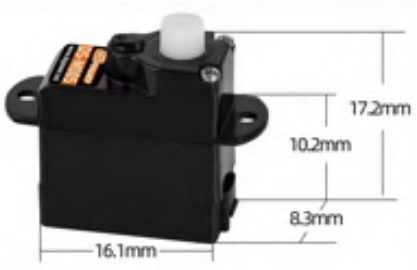


Localizing

The robots begin in a predefined structure where all positions are already known. Upon completion of their tasks, they will return to this initial structure. Positional data during operation can be derived from the known movements of each module. Additionally, to correct errors they are equipped with a gyroscope module, the MPU6050.

Power supply

The bots will rely on small batteries that will be continuously charged while attached to the system. The plan involves allowing current to flow through all bots in the system, creating a grid connected to a main power source. The small batteries should suffice so that in case the robot is disconnected it can walk back to the system.



Movement

The bot must move freely in 3D space using hinge-like motion; this is achieved with three servo motors. For compact controllable motors, I chose the DS-M005, one of the smallest servos available.

Operating Voltage	3.7~4.2V DC
No Load Speed	⩽0.08sec/60°at 3.7V (STD.)
	⩽0.06sec/60°at 4.2V (STD.)
Stall Current	⩽320mA at 3.7V (STD.)
	⩽350mA at 4.2V (STD.)
Max. Torque	⩾0.25kgf·cm at 3.7V(STD.)
	⩾0.30kgf·cm at 4.2V(STD.)
Pulse Width Range	500~2500μs
Operating Travel Angle	90°±10°
Max.Operating Travel Angle	180°±10°
Mechanical Limit Angle	360°
Weight	2.2±0.2g

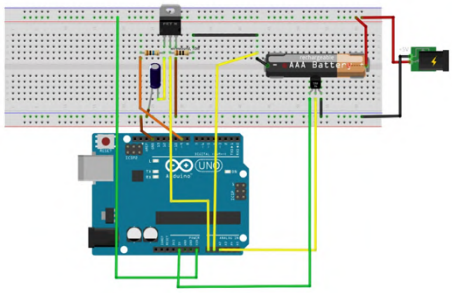
While its size is ideal, it requires a custom gearbox to boost torque, which is my priority over speed. Additionally, due to its low backward torque when off, I integrated a worm gear to prevent unintended leg movement, ensuring stability

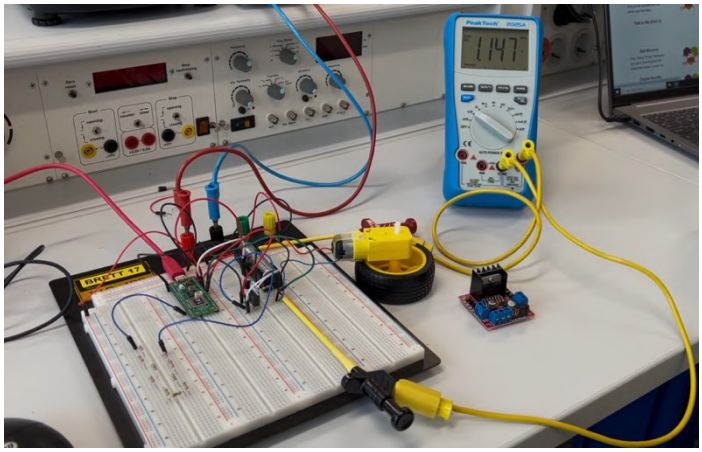
3. Project realization

3.1 Construction

A. battery charging

When the robot is connected to the grid, electrical current flows through contact points, which then needs to be efficiently converted and stored inside the battery. This process is made possible using a thermal sensor and a MOSFET. The thermal sensor plays a critical role in monitoring the temperature to prevent overheating, while the MOSFET is key in controlling the flow of electricity, ensuring that the charging is done safely and optimally, adjusting the charge as necessary to protect the battery's longevity and performance.

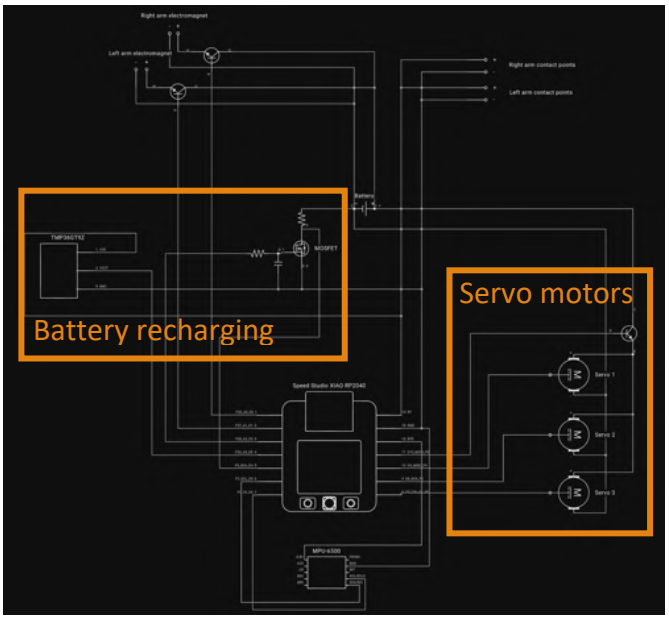


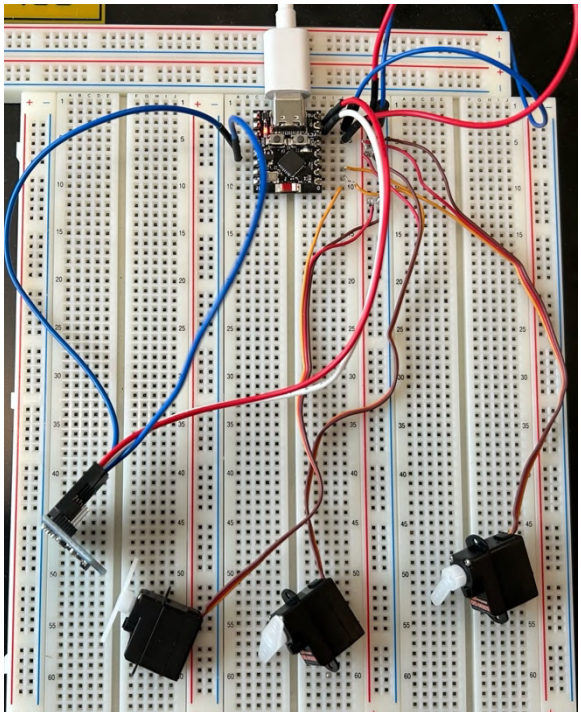


I first tested the circuit with a program that manages temperature and battery charge, confirming that it functioned as intended. I then simulated multiple scenarios: charging alone and charging while the motor was running, continuously monitoring battery voltage and external input. Based on the results, I confirmed that charging while running was a viable solution and that the entire circuit could be downscaled to the desired size.

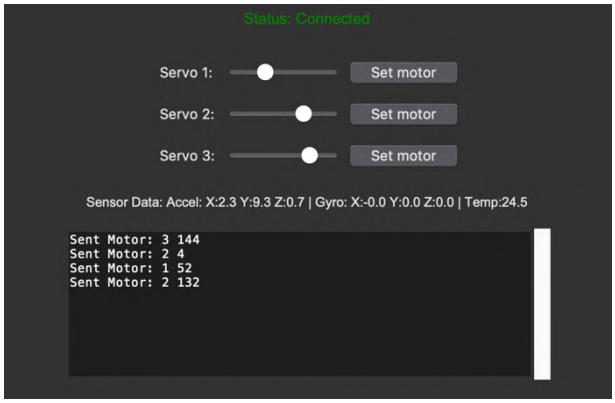
B. circuit diagram

I designed the entire circuit diagram using Flux, an online software with an extensive component library. I chose Flux for its user-friendly interface and integrated AI assistant, which helps detect potential errors. The core of the setup is the ESP32-C3. Upon receiving the correct sequence via Bluetooth, the microcontroller directly controls three servo motors, eliminating the need for motor drivers. Additionally, if movement is not required, a controllable transistor can disconnect all servos, preventing unnecessary power loss. To test the theory, I built the circuit on a breadboard and divided the programming into two phases: a C++ script flashed onto the



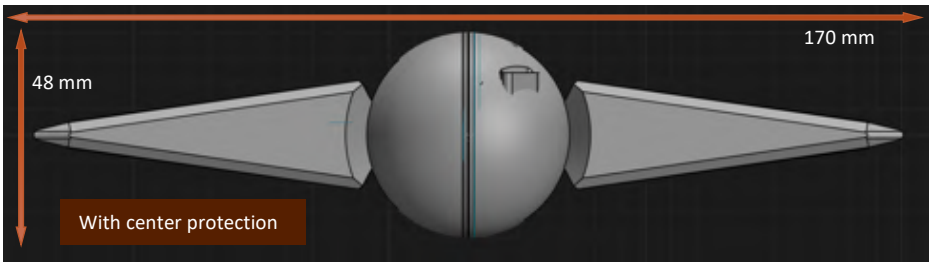


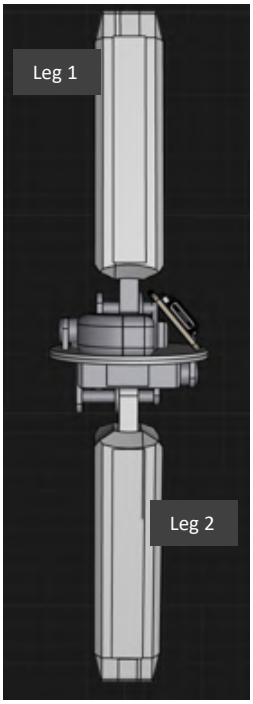
ESP32 via the Arduino IDE to read MPU6050 data and control the motors, and a Python script running on my laptop that uses the Bleak library to connect via Bluetooth. The simple interface allows for motor control and sensor data monitoring, with Bluetooth commands transmitted instantaneously over a range of up to 30 meters.



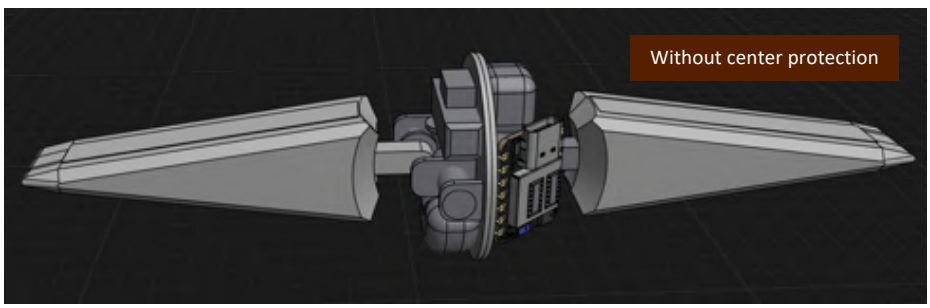
C. 3D model

First, I calculated the minimum volume which would allow for the possibility of the center sphere to have a radius of only 24 mm, and I aimed to get as close to this as possible. The limiting factors were the length of the motors, and I had to ensure enough space for cabling. Finally, I was left with the remaining space, which I used to maximize for the battery volume.



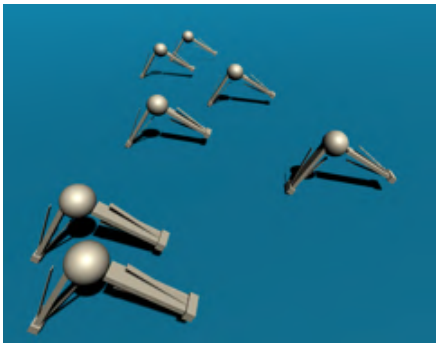


These are millimetre accurate 3D models made in Shapr3D considering the real size of all components in use. This image depicts a partially detailed 3D model showing the two halves of the center. Some cables

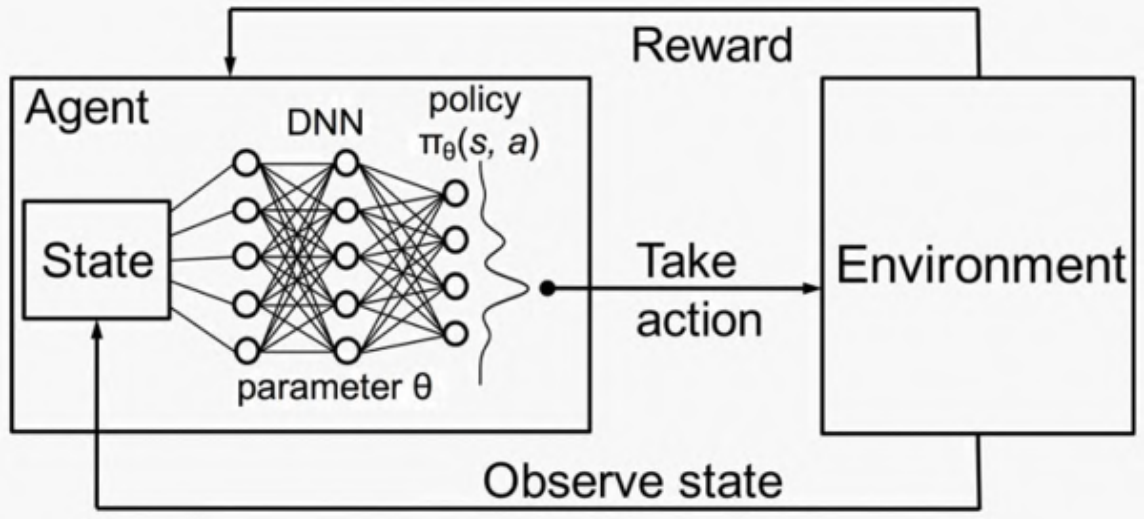


need to pass through the center bearing. However, this presents no issue since the center bearing never needs to rotate more than 180 degrees in either direction.

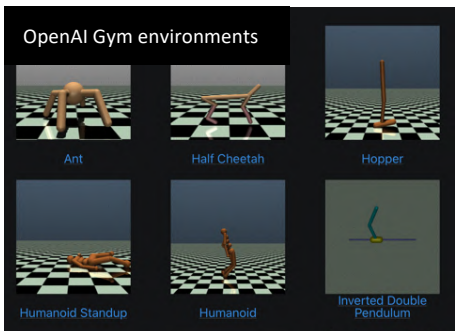
3.2 Using Deep Reinforcement Learning to Learn Walking



Initially, I attempted to implement walking through hardcoded movements, establishing a baseline for the robot's performance. However, this method resulted in slow movement speeds, prompting me to explore more dynamic solutions. So, I immersed myself in machine learning, drawing upon my experience with creating an AI module for the Chrome Dino game. Leveraging Jupyter notebooks facilitated training on Microsoft's Azure cloud computing service, while tools like TensorFlow, OpenCV, Keras, and OpenAI Gym enabled rapid development thanks to their extensive documentation. The success of the Chrome Dino AI project, which achieved near-endless running, served as a foundational step for developing the walking AI.



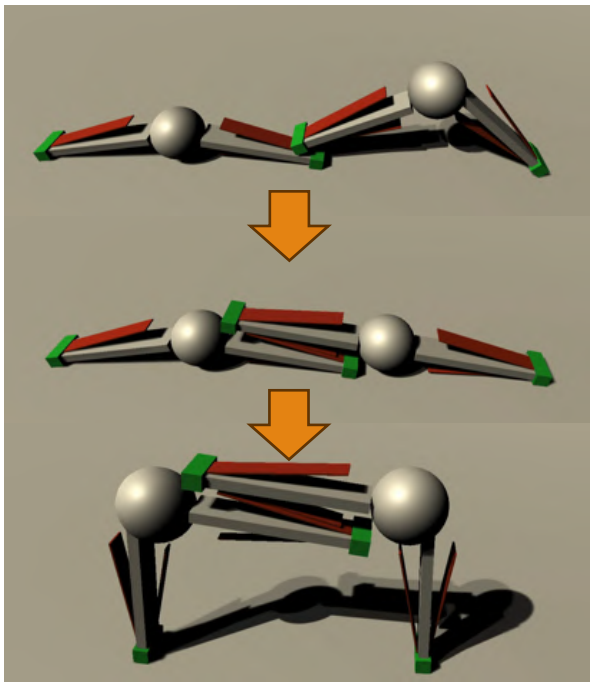
The neural network processes inputs, including the robot's center position and the current angles of its left and right legs, which are essential for determining stance and orientation, enabling precise movement control. The network's outputs dictate the desired angles for the left and right leg motors, directly influencing gait and balance during walking. A reward-based system encourages forward progression by assigning higher rewards for increasing the robot's x-position, promoting efficient movement. To enhance speed and efficiency, a 30-second time limit was introduced for walking tasks, forcing the AI to optimize both speed and effectiveness, ensuring a more refined walking pattern.



I created a custom environment within OpenAI Gym, which approximates real-world conditions with a simplified 3D model while allowing for extensive simulation and training. The agent's neural network model, constructed using Keras, processes inputs and predicts outputs to guide the robot's movements. Once I was happy with the module, I executed it with the real 3D model in Unity.

3.3 Base Simulations

The foundation of the entire system relies on the attaching operation between two modules. This fundamental process consists of the following steps:



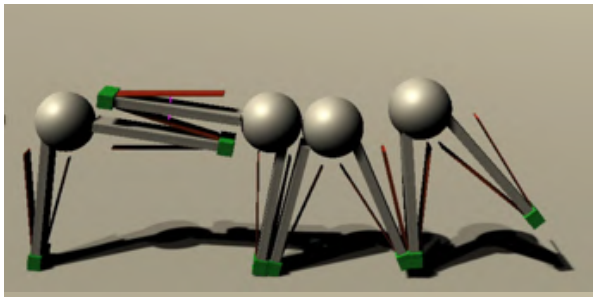
A module moves and aligns itself over the target module.

The module lowers itself onto the target module, ensuring precise alignment.

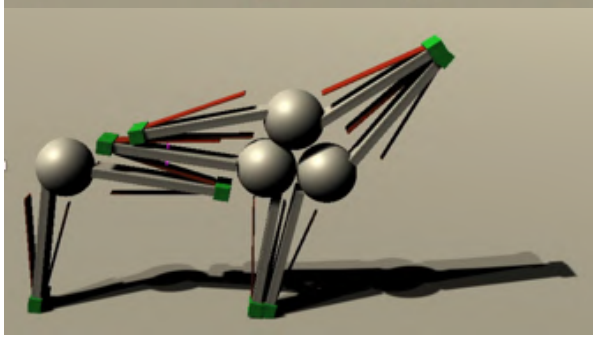
The magnets are activated, securely connecting the two modules and enabling them to hold firmly onto each other.

Another critical aspect of the system is the ability for modules to climb and stack on one another. This operation requires the assistance of an intermediate module to facilitate the elevation process. The steps are as follows:

The climbing module attaches itself to an intermediate module which lifts the climbing module above the target module, positioning it for attachment.



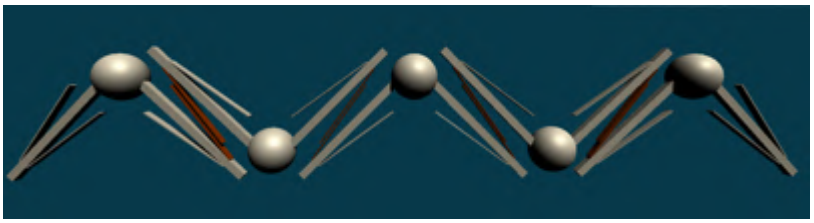
Once elevated, the climbing module connects to the target module. The intermediate module can then detach, completing the climbing process.



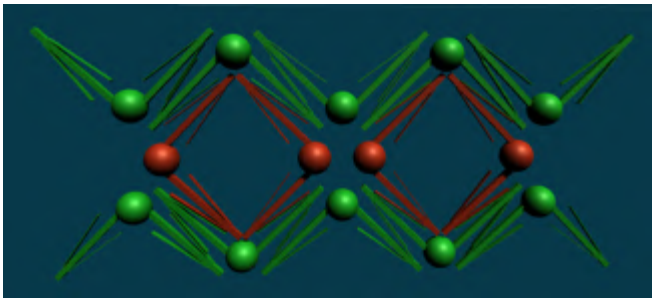
This simulation is essential for constructing more complex, multi-layered structures and demonstrates the system's versatility and scalability.

3.4 Structures

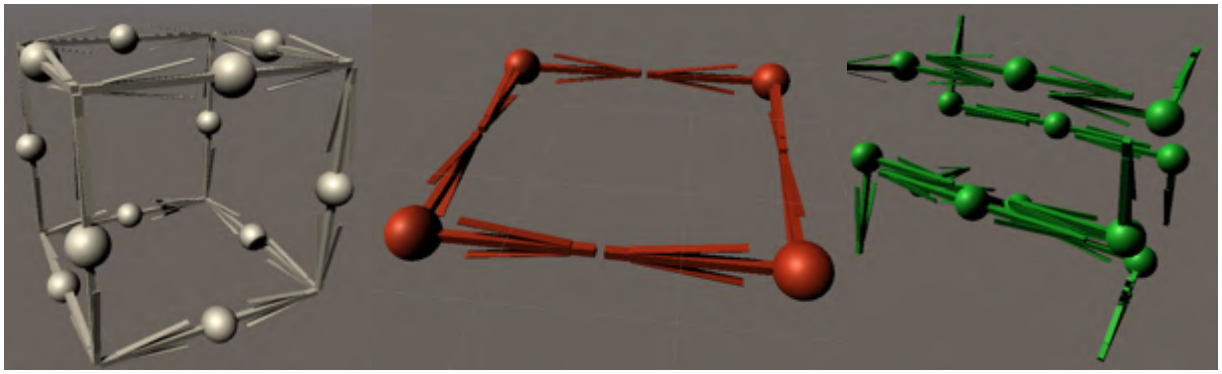
At first, I started with finding basic structures which could be stacked to make a bigger as show the following pictures. I am sure that with time I will find many others probably more stable and more efficient.



1D strip that can go on forever consisting of a sequence of flipped bots with both legs at a 45-degree angle.

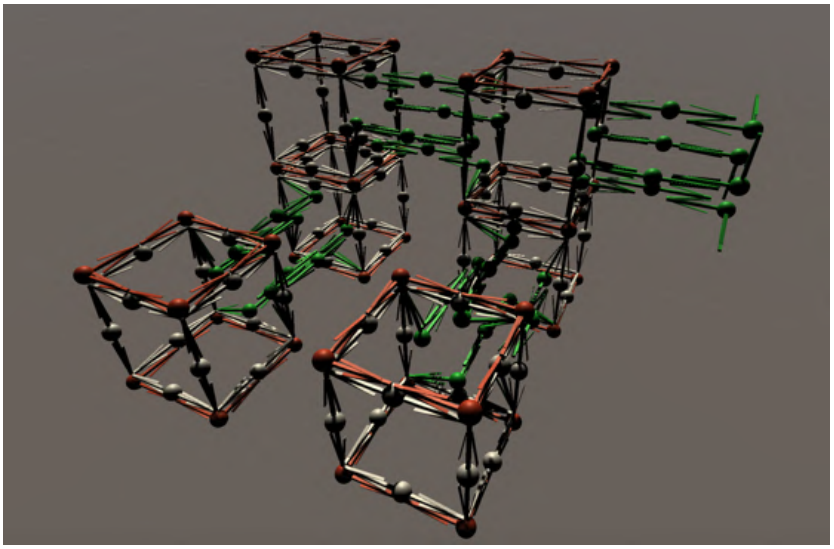


2D structure that can go on forever consisting of (green) the strips from before and (red) a part holding 2 strips together.



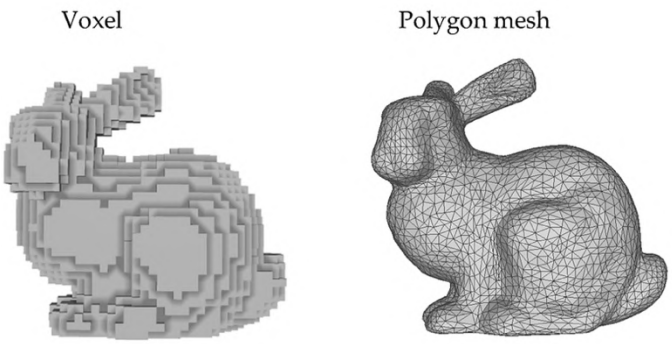
Going for 3 dimensional structures I tried to recreate basic geometry. Starting with a cube here in gray (only the outline) which wouldn't hold as no plates are touching. The red construct fixes this by holding 2 panels while the green connects the rest, and both acts as a bridge to the next Cube.

Combining those 3 elements it is possible to create a rigid structure that can stretch in all dimensions. This would for now be the ground structure of the system on which could be attached more specific structures.

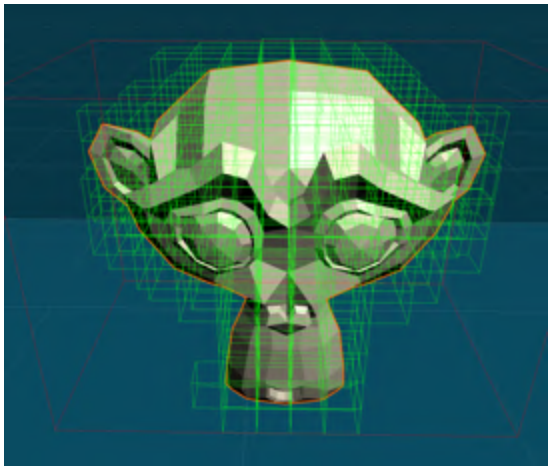


3.5 Voxelization Algorithms

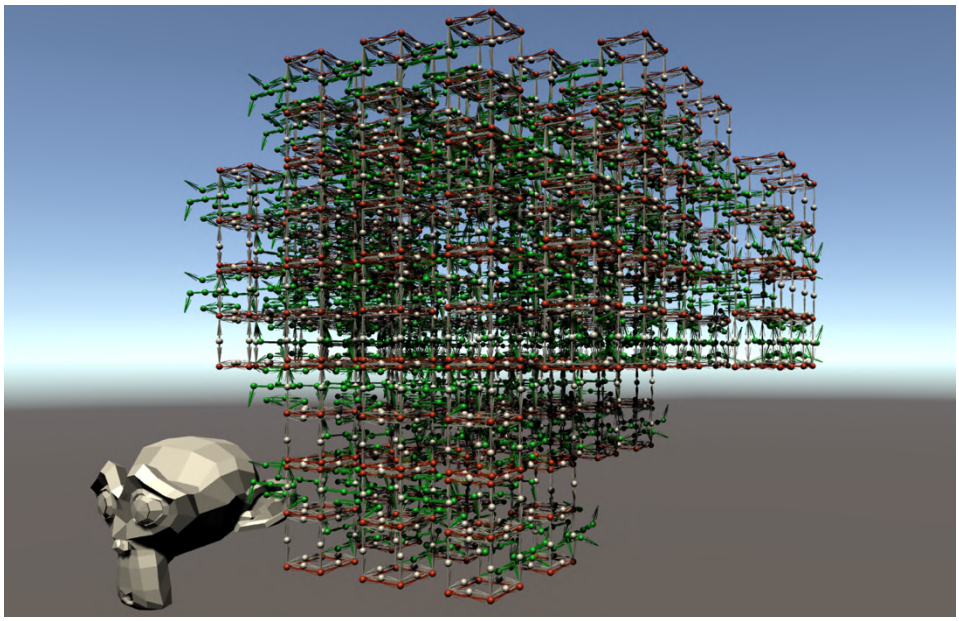
Voxelization algorithms convert geometric objects into a discrete grid of voxels (3D pixels), capturing the 3D structure in a regular, grid-based format. It is like if you forced a mesh to conform to a 3D grid. Unlike polygons that define surfaces through vertices and edges, voxels represent volume, making them ideal for applications requiring volumetric analysis or rendering.



The key aspect of those algorithm is Occupancy Evaluation which is determining whether a voxel is inside, outside, or on the boundary of the model. The detailing of the voxelated object can easily be changed by adapting the size of a single voxel of course this will make the program more resource intensive.



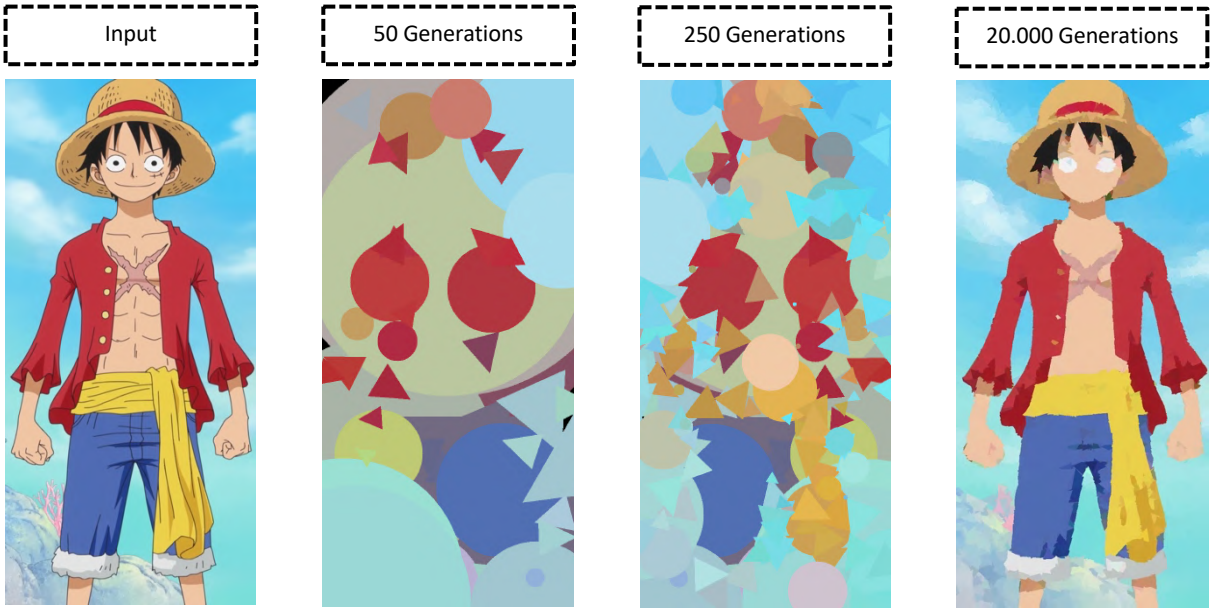
Here I am using Suzanne the blender monkey for my test. The algorithm creates a list which than spawns a block of modules for each voxel. This voxelization tactic allows me to replicate any 3D model I feed it. As such it becomes easier to mass control the worker bots. For example, act as a support pillar to protect a collapsing bridge or ancient ruin. It is also useful to have a 3D representation of an object your enterprise is currently working on. The stability isn't always assured, and I propose to add support like the ones used when 3D printing.



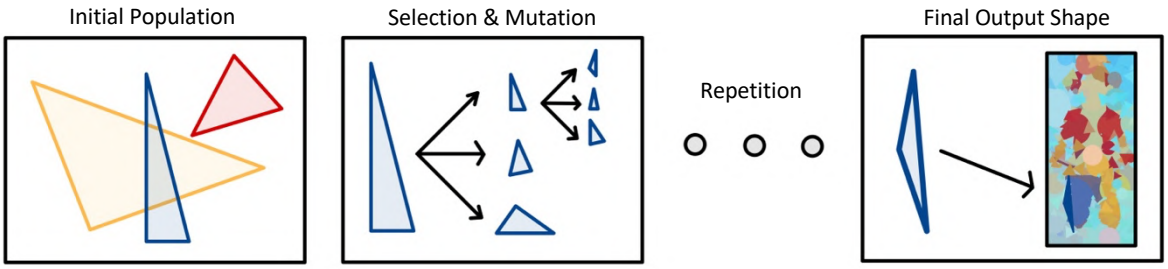
3.6 Genetic Evolution Algorithms

Voxelization enables fast 3D model reproduction but lacks stability and organization. To improve the system, I let it evolve and optimize itself using principles from Darwinian evolution and survival of the fittest.

To break it into manageable stages I began with a genetic algorithm that reproduced a 2D image using simple geometric shapes. This stepwise approach allowed me to find more documentation, even most existing research was unsuitable for my specific problem.

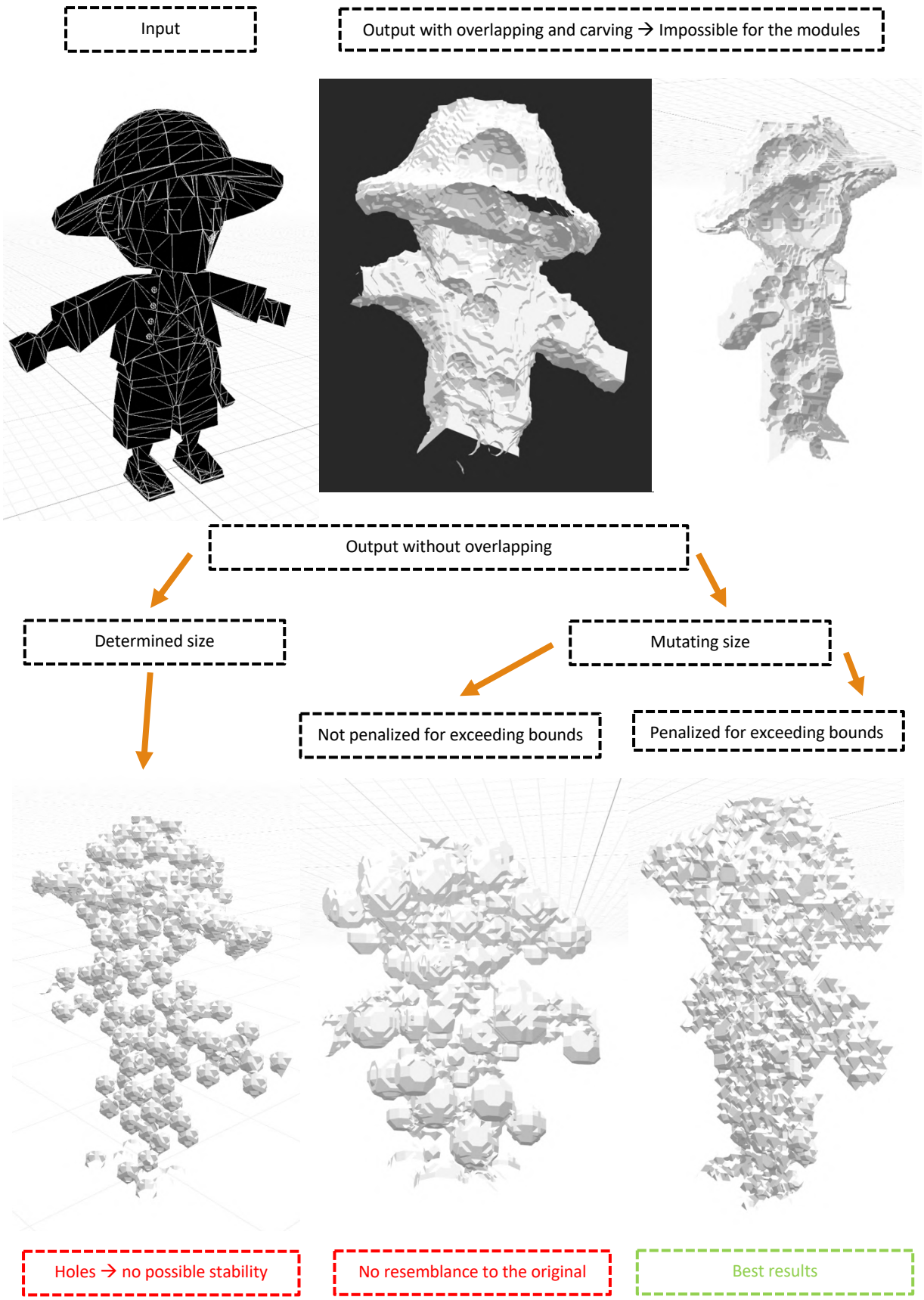


- The **initial population** consists of randomly generated solutions. In my case, X triangles with random parameters: the coordinates of the three corners, RGB colour values, and opacity.
- The **fitness function** measures how well each solution approximates the desired outcome. In my case, it evaluates the pixel-wise difference between the generated and target images by computing the Euclidean distance in RGB space.
- The fittest subjects are selected as parents, passing on their traits to generate **offspring** that ideally inherit the best characteristics. Additionally, I introduce **mutations**, low-probability random alterations, to explore the solution space effectively.



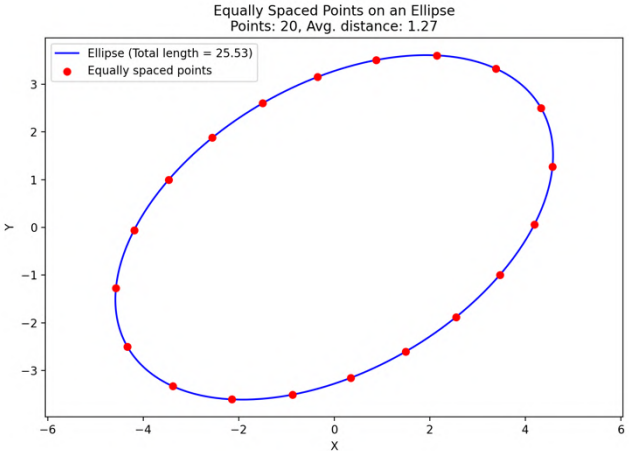
This process is repeated for N shapes while continuously reducing mutation rate to optimize for large shapes at beginning and finer detail later.

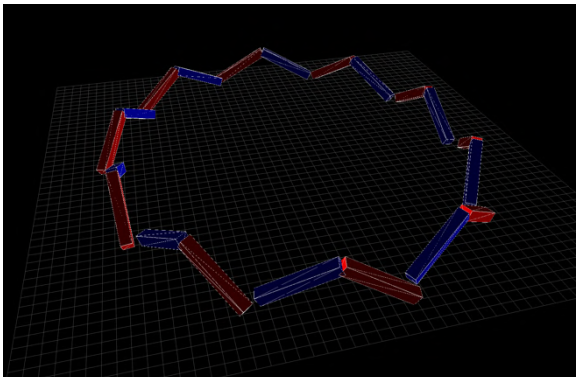
I then switched to reconstructing 3D models using spheres. I tested numerous options, which impacted the results and the potential applications for the modular robotic system.



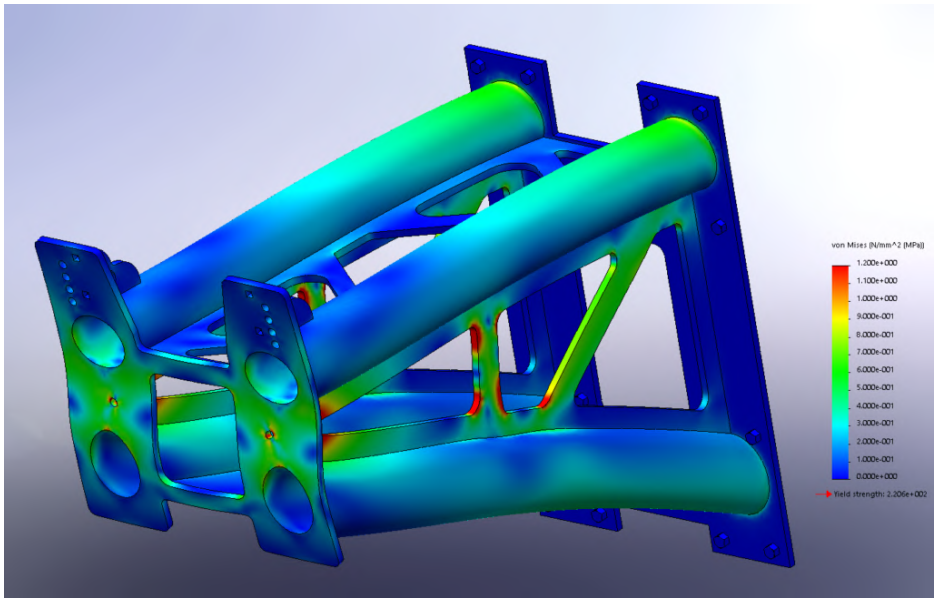
To create useful structures, I identify two main goals: achieving the best visual representation or maximizing stability. In the first case, the outer skin is the priority, and the structure must support it while maintaining integrity. In the second case, the focus is solely on stability, making the system highly practical.

In both cases, the algorithm must first determine the ground layer to establish a valid starting point. Then, I set constraints to allow only physically feasible configurations. To determine the modules position along the ground border my algorithm calculates the cumulative arc length, inverts it with interpolation to map desired arc lengths back to t values, and then divides the perimeter into equal segments to compute the corresponding (x, y) coordinates.





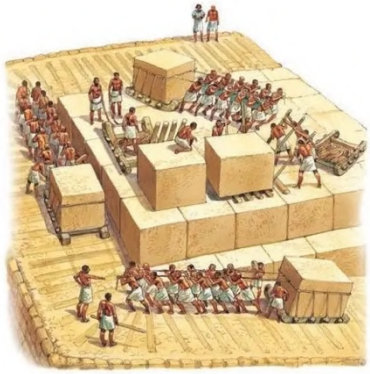
Each point is then replaced with a module tangentially aligned so that the legs converge at their centers with the required angles. Next, I aimed to apply my genetic algorithms; however, developing an effective fitness function remains a challenge. This function must assess structural stability by ensuring that all modules are interconnected and can withstand stress tests. This rigidity analysis is computationally intensive, making it impossible to run for every mutation of every generation. Consequently, this aspect remains a work in progress.



3.7 Centralized Mass Control

To enable coordinated movement of the bots, mass control algorithms are essential. The headmaster, a nearby computer, undertakes all the necessary calculations and issues commands via Bluetooth, providing each module with instructions.

This process unfolds in three steps:

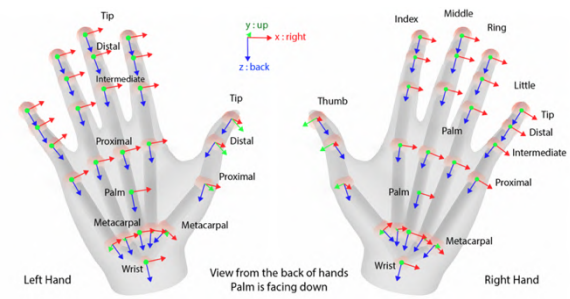


- **Ground Structure Construction:** Similar to the methods employed by the ancient Egyptians when constructing the pyramids, the bots ascend from the sides and take their positions. Movement within the system is possible by sequential engagement and disengagement of the metal plates. If certain areas are too confined or require impossible rotations, additional worker modules can assist in advancing further.
- **Special Modules:** Specialized modules or tools are transported to their designated locations either by autonomously navigating through the structure or by being carried by multiple worker modules.
- **Constant Outer Layer Reorganization:** Throughout the task execution, there is ongoing reorganization of the outer layer to accommodate evolving requirements.

3.8 Virtual Reality Simulation with Hand Tracking & Passthrough

As in the film “big heroes 6” the microbots are controlled with their brain I wanted to try and achieve something similar.

Drawing from my experience developing a hand-tracking VR game, I intend to use this remarkable technology to interact with a simulated version of the system. This approach allows for faster prototyping and visualization. Unity simulations can also be run on VR headsets, albeit with some limitations. Therefore, my goal is to create an interactive environment for users to explore and become familiar with the system's capabilities. This approach emphasizes interaction, enabling players to control the system using hand gestures and allowing it to take shape as desired.

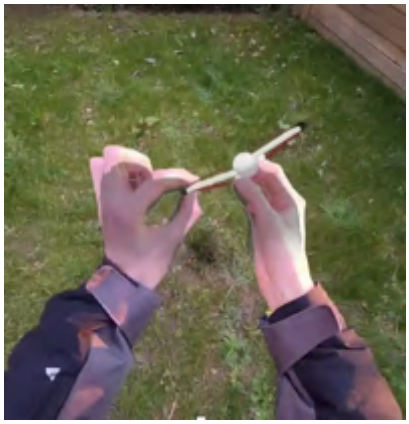


Hand gestures or poses are defined by the position and rotation of various parts of the hand, such as fingertips and knuckles. Additionally, with the pass-through feature available in the new Meta Quest 3, the simulation can appear to take place in the user's actual room, enhancing the sense of realness.

Unlike VR games, where environments mainly enhance immersion, this project's surroundings serve a functional role. A plain white room would lack context and could induce claustrophobia or anxiety due to its emptiness. Moreover, for a system interacting with the real world, visualizing it in its actual environment is crucial for understanding scale, functionality, and practicality. To address this, I implemented passthrough, blending real and virtual worlds by overlaying the system onto the user's actual surroundings. This ensures a more intuitive, context-aware experience, allowing for better evaluation and interaction.

The process begins with the VR headset generating a 3D mesh of the surrounding environment. This mesh acts as a rough approximation of real-world geometry, providing the system with essential spatial data. By combining this information with coloured passthrough, users can view the system superimposed on their physical environment, making it easier to assess how the system interacts with its surroundings and how it would function in real-world scenarios.

The initial implementation focused on enabling users to inspect the 3D model in detail, with the ability to scale it up or down for close examination. Using intuitive hand gestures, such as pinching, the system detects when a user attempts to grab an object. When two hands are used to grab, the object dynamically scales based on the distance between the two grab points, offering precise and natural control over size adjustments.



Building on this foundation, I added functionality to move individual modules and attach them together. This feature significantly enhances the workflow, allowing users to quickly iterate through various structural designs. It also provides a deeper understanding of the assembly and mechanics of the structure within a fully interactive 3D space. The combination of scaling, manipulation, and modular interaction ensures a highly efficient and immersive design process, bridging the gap between virtual prototypes and practical implementations.

These images are screen captures from my application during testing in my garden. The gray hands depicted represent the headset's approximation of hand positions.

3.9 Testing & Problems

During the realization of the project, I encountered numerous challenges and encountered a multitude of programming bugs.

A) Center bearing

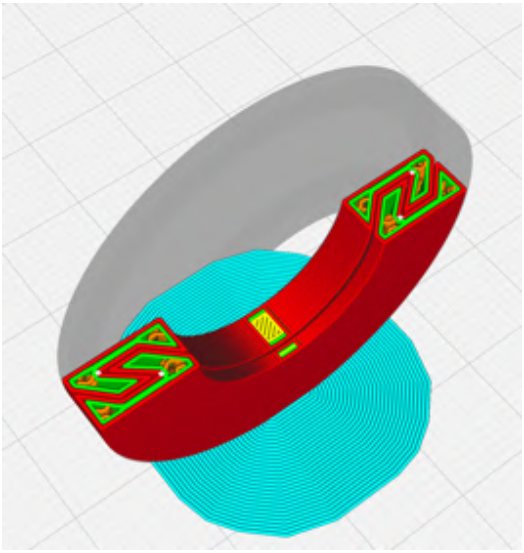
While testing the simulations, Unity's physics-accurate engine allowed me to assign breaking forces to the robot's joints for stress testing. Early in the process, I identified a significant issue: the center bearing could not adequately support any load. Even before reaching the breaking point, its rotation became compromised under strain, severely impacting functionality. A possible solution could come from the advancements in 3D printing: in-place printing.

What is In-Place Printing?

In-place printing refers to the process of designing and manufacturing moving parts or assemblies directly within a single print, without requiring post-assembly. The components, such as bearings or joints, are printed in their final operational configuration, with appropriate clearances designed into the model. This method eliminates the need for assembly, improves part integrity, and allows for the creation of complex mechanisms that would otherwise be difficult or impossible to produce.

Implementation

Using in-place printing, I designed a bearing that is fully integrated within itself, creating a highly resistant structure. This design ensures that the bearing can withstand significant loads without sacrificing rotational functionality.



The images provided show a cross-section of the bearing as visualized in Ultimaker Cura, the 3D printing software I am using.

Additionally, for future iterations where reduced friction is necessary, ball bearings could be incorporated into the design to further enhance performance and efficiency. This evolution would make the system even more robust and adaptable to a wider range of applications.

B) Center of gravity

During the initial design phase, I intentionally aimed to position the center of gravity (CoG) perfectly at the geometric center of the sphere. This was crucial because a balanced CoG ensures stability during motion and reduces the likelihood of the robot tipping or experiencing uneven stress on its joints. A well-centered CoG is particularly important for modular robots like this, as it allows for smoother and more predictable movement, enabling the robot to handle complex tasks with greater precision.

However, I did not initially account for the dynamic nature of the robot's complex movements. During operation, certain motions or configurations can cause the CoG to shift outside the intended center, potentially compromising stability and efficiency. This unforeseen challenge highlights the need for additional refinements in future iterations. Possible solutions may include redistributing internal components or adding counterweights.

C) Energy distribution

Another critical challenge centered on the distribution of current within the system. Modules further from the initial power source would risk getting not enough voltage. To address this, I propose three potential solutions that can be integrated:

Multiple Power Sources: Introducing multiple power sources throughout the system could mitigate voltage drop issues. For instance, additional power sources could be installed on the ceiling to supplement charging. However, this approach may increase complexity, which is not ideal.

Selective Charging: Implementing a system to deactivate charging for bots that are stationary once their batteries reach a certain threshold could help optimize power distribution. This ensures that energy is prioritized for active modules, enhancing overall efficiency.

Distributed Battery Modules: Introducing specific modules equipped with batteries distributed throughout the system could provide localized power sources. These battery modules can be strategically placed to supplement power where needed.

3.10 Results

In the point 1.4 motivation I explain 3 problems of current modular systems.

1. Thanks to the motors the module has complete freedom and can walk on a surface in all directions. It can also form complex 3D structures where specific angles are required. More details on point 2.2, 3.1 and 3.2.

2. Going initially for the theoretical proof allowed me to go faster towards the control and structural algorithms. Additionally, the 3D voxelization allows for easier control with the ability to give a 3D model as base input. More details on point 3.5 and 3.6.

3. During the entire process I tried to minimize size which worked partially. The main constraint being the length of the motor. I believe this can still be reduced if designing a custom motor. Comparing it to other modular system is difficult as no size data is always provided. A size of (48 * 140 mm) is decent as I use cheap components everyone can buy online.

4. Extras

4.1 Conclusion

I embarked on a quest to bring to life a concept from a fictional film, delving into the realms of modular robotics and self-assembling systems. Beginning with an exploration of these concepts, I eventually achieved a significant milestone: the theoretical demonstration of constructing a worker module. Moreover, I successfully tackled the challenges of controlling larger systems through voxelization algorithms. Lastly, using deep reinforcement learning, I enabled the module to walk faster than I could ever be achieved through traditional hardcoded methods.

Throughout this journey, I harnessed knowledge of important technologies, including 3D modelling, electronics, programming, machine learning, and virtual reality. These tools were instrumental in realizing the vision and pushing the boundaries of what is possible with modular robotics.

As of now, the mission has been a successful, and the potential use cases for such modular robotics are limitless. From everyday tasks like cleaning and tidying one's house to more critical applications such as supporting a collapsing bridge, the project stands ready to tackle any challenge thrown its way.

4.2 Future Plans

My next significant milestone is the final assembly of a worker module.

I already tested all the electronic circuits but once a prototype is completed, I will be able to do physical tests to compare reality with the simulations and adjust the algorithms. I also aim to adapt the AI for turning, and navigating rough terrains, and identifying more efficient and stable structural designs.

In summary, while my adventure with this project is still ongoing, the progress made thus far has been promising, and I have successfully demonstrated the feasibility of the main concept.

5. Appendix

5.1 References

Official Documentations & forums:

- Oculus SDK
- Unity 3D
- OpenAI Gym
- TensorFlow
- Wiki.Seedstudio.Com

Articles & papers:

1. Ronan Hinchet, Velko Vechev , Herbert Shea & Otmar Hilliges, DextrES: Wearable Haptic Feedback for Grasping in VR via a Thin Form-Factor Electrostatic Brake, aitlab

2. Mariella Moon, Echolocation could give small robots the ability to find lost people, engadget

3. Davide Cavagnino & Marco Gribaudo, Discretization of 3D models using voxel elements of different shapes, researchgate

4. Park Chan-Il & Cho Do-Hyun, Comparison of Dynamic Characteristics of Spur Gears and Helical Gears, KoreaScience

5. Jianglong Guo, Jinsong Leng & Jonathan Rossiter, Electroadhesion Technologies for Robotics: A Comprehensive Review, ieeexplore

6. Jason Poel Smith, Create an Arduino Controlled Battery Charger, allaboutcircuits

7. Celera Motion, A Guide to Robot Joint Design, azorobotics

8. Mengran Gao, Ningjun Ruan, Junpeng Shi & Wanli Zhou, Deep Neural Network for 3D Shape Classification Based on Mesh Feature, MDPI

9. Nicholas Renotte, Complete Machine Learning and Data Science Courses, youtube

10. A. Utsumi, J. Ohya, Multiple-hand-gesture tracking using multiple cameras, ieeexplore

11. Sudharsan Ravichandiran, master reinforcement and deep reinforcement learning using OpenAI gym and tensorFlow, google scholar book

12. Zushi Tian, Ye Tian, Hailong Ye, Xianyu Jin & Nang.o Jin, VOX model: application of voxel-based packing algorithm on cementitious composites with 3D irregular-shape particles, springer link

13. Bronson Zgeb, Simple Mesh Voxelization in Unity, bronsonzgeb.com

14. Bram Lambrecht, Voxelization of boundary representations using oriented LEGO®plates, semanticscholar

15. Yuxi Li, Deep Reinforcement Learning: An Overview, arxiv

16. Sayon Dutta, A beginner's guide to designing self-learning systems with TensorFlow and OpenAI Gym, google scholar book

ChatGPT:

1. Occasionally, after writing my own text, it was used to correct grammatical errors and enhance the fluidity of the text, allowing for quicker revisions.

2. To find synonyms and enrich the vocabulary or to modify certain expressions.

At no point did ChatGPT generate new text with information that I hadn't provided or restructure my texts. It solely rephrased existing content. The output was then reviewed and refined. Thus, ChatGPT did not directly influence the content of my project but facilitated the creation of a better report than I could have achieved alone in such time.

5.2 Image sources

If not indicated the images in this report were self-made.

Page 2: Big Heroes 6 microbots: sidefx.com
Modular robot left: newscientist.com
Modular robot middle: staff.aist.go.jp/e.yoshida/test
Modular robot right: news.mit.edu/2019

Page 3: Furniture modular robot: actu.epfl.ch/news
Modular robot left: robotics247.com
Modular robot right: spectrum.ieee.org

Page 4: Beehive: apis-donau.com

Page 5: Electromagnet: aliexpress.com
ESP32-C3 with pins: sigmdel.ca/michel/ha/esp8266/super_mini_esp32c3_en
MPU6500: motorobit.com/mpu6500-6-axis-acceleration-and-gyro-sensor
Battery: sharvielectronics.com/ lipo-rechargeable-battery-3-7v-250mah
Servo Motor: aliexpress.com

Page 6: Battery recharger: allaboutcircuits.com/projects

Page 8: Chrome Dino: play.google.com
Reinforcement Learning Graph: medium.com/@vishnuvijayanpv
OpenAI GYM environments: gymlibrary.dev

Page 11: Rabbits: mdpi.com/1424-8220

Page 12: Character: one-piece.com
3D input Character: sketchfab.com/3d-models/luffy-one-piece/D7xtreme

Page 14: Stress test: dcwhite.co.uk/fea-stress/

Page 15: Egyptian pyramid: pinterest.com
Hands: Godotengine.com

Page 16: Passthrough room scan: developers.meta.com