University Institute of Lisbon

Department of Information Science and Technology

Co-evolution of Morphology and Controller for a Robot

Ivo Manuel Caeiro da Silva

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Computer Engineering

Supervisor

Assitant Prof. Dr. Sancho Moura Oliveira, assistant professor

ISCTE-IUL

Co-Supervisor

Assistant Prof. Dr. Anders Lyhne Christensen, assistant professor

ISCTE-IUL

September, 2018

"I started concentrating so hard on my vision that I lost sight."

Robin Green

Resumo

Os algoritmos genéticos são inspirados pelo processo de seleção natural que existe na natureza. Este processo leva espécies a evoluir e adaptar-se ao meio ambiente envolvente, com as espécies mais aptas reproduzindo, levando a que novas gerações possam tirar um melhor proveito do ambiente que as rodeia. Este tipo de processo pode ser utilizado na robótica evolucionária para evoluir controladores capazes de resolver tarefas de forma a evoluir morfologias para uma finalidade específica, tais como andar, nadar, agarrar objetos, entre outros.

Garras robóticas são utilizadas na maioria das fábricas, assim como noutros locais de trabalho tais como hospitais e laboratórios. Podem ser utilizadas em tarefas como agarrar/mover objetos, pintura, cirurgias, entre outros usos. São, portanto, um caso de estudo com várias possibilidades que se prestam à evolução de morfologias através de algoritmos genéticos.

Nesta dissertação, exploramos a geração de morfologia através de algoritmos genéticos. Utilizando garras como o nosso caso de estudo, conseguimos gerar garras capazes de agarrar e levantar um objeto. Para evoluir essas garras, criamos um ambiente simulado onde cada garra seguiu um script com instruções para agarrar o objeto e, em seguida, mover para cima. No total, 120 garras diferentes foram geradas nestas experiências. Dessas garras geradas 120, 28% foram capazes de capturar e levantar um objeto com êxito.

Após a conclusão do processo de avaliação, experimentamos as garras em cinco cenários diferentes para testar a sua robustez. Nesses cenários, as condições iniciais em que os objetos começam eram diferentes das do processo de avaliação.

Palavras-chave: Garra, Geração de morfologias, Algoritmos Genéticos.

Abstract

Genetic algorithms are inspired by the process of natural selection that exists in nature. This process is what leads species to evolve and adapt to their surroundings, with the fittest species reproducing, leading to new generations that can take advantage of their surroundings better than before. This type of process can be used in evolutionary robotics to achieve controllers that are able to solve specific tasks to evolve morphologies for a specific purpose such as to walk, swim, grasp objects, among others.

Robotic grippers are used in most factories nowadays, as well as in other workplaces such as hospitals and laboratories. They are used in tasks such as grabbing/moving objects, painting, surgeries, among many other uses. Grippers are therefore a case study with several possibilities that lend themselves to evolving morphologies through genetic algorithms.

In this dissertation, we explore morphology generation through genetic algorithms. Using grippers as our case study, we were able to generate grippers capable of grabbing and lifting an object. To evolve these grippers, we created a simulated environment where grippers followed a script with instructions to grab the object and then move up. In total 120 different grippers were generated in these experiments. Out of those 120 generated grippers, 28% were able to grab and lift an object successfully.

After the evaluation process was completed, we experimented with the grippers in five different scenarios to test their robustness. In these scenarios, the object's starting conditions were different from those in the evaluation process.

Keywords: Gripper, Morphology Generation, Genetic Algorithms.

Acknowledgements

I would like to start by thanking my supervisor Professor Sancho Oliveira for guiding me through the process of building this thesis. His opinions and feedback were crucial in moving the thesis in the right direction.

I would also like to thank my co-supervisor Professor Anders Christensen whose feedback led to a more coherent and complete thesis.

Thanks to João Sousa from Vitruvius FabLab for helping with building and modeling the 3D pieces we use when generating the final gripper.

Thanks to my parents, Mário Silva and Cidália Silva, whose support and love made me carry on and not give up, even in the hardest moments. Not understanding the theme of this thesis didn't stop both of you from trying to help me along the way, and for that I am thankful.

Thanks to my friends for being there, especially Diogo Vicente and Nuno Coelho, whose support, words of encouragement, and long conversations about everything helped me come this far.

To all those who contributed in some way to this journey, I sincerely thank you.

Contents

List of Figures

Chapter 1

Introduction

Industrial robots are typically used in factories. They have been replacing humans for the last decades in completing repetitive tasks that can be automated. What makes these robots valuable in the workplace is the speed and precision with which they are able to perform tasks and the reduction of costs.

These robots are present in different industries and while they have different purposes, their core remains the same; the robot must be programmable and must be able to move or rotate in at least two axis. Robotic arms are programmed to fit the designated task, and the same happens to the robotic arms degrees of freedom. Depending on the task, the arm will have more or less freedom of movement along the three axis (x, y, z), and the rotation along the axis (yaw, pitch, roll).

These robotic arms are versatile, and as such they can be found in various factories from different industries. A common use for robotic arms is in automotive factories, where they are used in very different ways. In this industry, the robotic arm is used to weld, assemble and paint car parts.

Robotic arms have at their ends a robot part designated as an end-effector. This piece can be modified to adapt to the task. For example, a robotic arm used to weld needs to have a welding piece attached at its end to perform the task, while a robotic arm whose task is to paint parts, needs a tool that can shoot ink, for example a spray gun to paint. For tasks directed at grabbing and moving objects, a gripper is a more fitting tool. The grippers are usually used to move objects from one point to another, or to hold the object still while another tool, or worker works on it.

Creating a gripper is a complex task, which is why such task can be split into different categories for instance, heavy/light objects, small/big objects, or even long/short objects for the gripper to grab. By splitting the possibilities into single objects for the gripper to grab, gripper creation becomes a more manageable task. By doing this, specific grippers are created to grab specific objects.

However, nature's own evolutionary process led to the generation of a versatile gripper known as the human hand. This gripper, despite being versatile, cannot grab objects as easily as a robotic gripper built for that purpose. Our hands required years of training to grab objects, and some objects we may never be able to grab due to our hands properties, or the objects surface properties.

Unlike our hand that can only grab objects by grip and force, robotic grippers can be divided into four categories, depending on which technique they use to grab objects:

- Impactive, one of the most common types of grippers that use the same techniques as our hands to grab objects (force and grip);
- Ingressive, these types of grippers use pins or needles to penetrate the object in order to grab it;
- Astrictive, which use suction techniques (vacuum, magnetism, electro adhesion) to grab the object;
- Contigutive, these ones require a type of glue to grab the object.

The purpose of this dissertation is to evolve a gripper capable of grabbing objects independent of their size or rotation, and for that we chose to simulate the impactive type of gripper to conduct our experiments in evolving grippers.

The impactive gripper is one of the most basic types of gripper, and one that emulates our hands to some extent in the actions it takes to grab objects. This type of gripper applies an amount of force to grab an object, unlike for example, the ingressive gripper that uses pins or needles for the same task.

To run these experiments in order to evolve grippers, we used genetic algorithms. This type of algorithms were first introduced by Holland (1962) and are based around biological processes observed in nature, as such, it makes sense to try and experiment with them to evolve the impactive gripper's morphology so that we can get solutions that followed the same principle as our own hands.

Genetic algorithms are commonly used in evolving morphologies and controllers. With genetic algorithms the simulation starts with a population of randomly generated individuals, with each individual trying to complete the task and being attributed fitness points depending on how well it performed the task. The best individuals are then chosen to start a new population by being cloned. Those clones have a chance to suffer mutations. These mutations allow for different solutions to emerge within the simulation. This process repeats until the number of generations is reached.

Due to their versatility and accessibility, impactive grippers were chosen to conduct our study on evolving morphologies through genetic algorithms.

To conduct this study, we ran simulations of a gripper that had its number of blocks, joint connections, and block morphology evolved through Genetic Algorithms. In these simulations, we simulated the gripper grabbing different objects in different sizes. Except for the sphere, all other objects (capsule, cylinder, rectangle) were tested in different angles of rotation.

1.1 Objectives

The main objective of this dissertation is to explore gripper morphology generation through genetic algorithms so that the generated grippers can complete tasks such as: grabbing an object independent of its rotation; grabbing the same object independent of its size; grabbing different objects. To achieve this, we start by first evolving morphologies that follow a script detailing the behaviour the gripper should follow at each point of the simulation to complete the task (when the gripper should move up and when the gripper should grab the object).

After the main evolution was conducted, we experimented with the evolved grippers in five different scenarios. These five scenarios all had noise introduced in them. This noise was a random deviation in the positioning of the object by one millimeter, as well as one-degree deviations in rotation of the object.

The scenarios are different from each other, with the first one being just the introduction of random noise in each of the angles used during the main experiment. The second scenario tests if the grippers can grab and lift different solids, and not just the one they were originally evolved to grab and lift. In the third scenario, one hundred random angles are generated for the object the gripper has to grab and lift. With this experiment, we analyze if the grippers can grab the object in angles different from the ones it originally evolved to grab and lift. The fourth experiment increments the object size for each gripper, starting from the smallest size used in the main experiments, up until the maximum size used. In the final scenario, the grippers try to grab and lift different sizes used during the main experiment.

1.2 Research Questions

- How to evolve the morphology of a gripper given a task;
- Are genetic algorithms suited for evolving the morphology of a gripper;
- Do evolved grippers show an homogeneous morphology;

By answering these questions, we expect to advance the research of gripper block morphology generation further.

1.3 Structure of the Dissertation

Starting with Chapter 2, other authors' ideas and methodologies that are relevant to this dissertation are presented and discussed at length. In Chapter 3, we detail the methods used to create the simulator. In Chapter 4, we discuss the results gathered from the main evaluation experiments. In Chapter 5, we discuss the results of the five scenarios experimented with after the main evolution. And we conclude this study in Chapter 6 by presenting a final assessment about the results and future work.

Chapter 2

State of the Art

One can consider that a robot is composed by a controller and its morphology. A controller is the "brain" of the robot. Much like our brains, it receives signals and acts upon them. It is the controller that controls a robots behavior. This behavior can be walking forward, picking up objects, amongst many other behaviors. The morphology is the body of the robot. It can range from a humanoid form to one that is solely built to complete a specific task, such as just an arm.

Controllers and morphologies can be generated and evolved through different algorithms with the most popular being genetic algorithms and neuroevolution. To evolve the controllers and morphologies, we must experiment with them in a simulated environment that allows us to simulate our task, which can be simple or detailed, depending on what we intend to simulate. By iterating over this process, our controllers and morphologies get better and more efficient at solving the proposed task. This process can be repeated until a maximum is reached, or until we reach the maximum number of generations.

2.1 Evolving Controllers and Morphologies

Evolution of controllers or morphologies can be described as the evolution of the brain or the body parts of a robot. Evolutions start with randomly generated populations that then have to complete the given task. As the evolution progresses, the controllers or morphologies should also get better than those that came before, evolving to better adapt to the given task.

Both the controller and morphology can be evolved through evolutionary algorithms. In the case of the controller, it can be programed to just complete a task, if the focus is in different morphology generation. In the case of morphology, we can use an existing morphology if the study focuses only in evolving controllers.

To evolve a controller or morphology, there are many techniques available, it is up to the researcher to choose which of those techniques fits the task at hand. In this study we will be focusing on evolving the morphology.

This section details some techniques used to evolve controller and morphology. These techniques make use of algorithms known as *"Evolutionary Algorithms"*, detailed in section **2.2**.

2.1.1 Automatic generation

Sims (1994) created a system where 3D creatures can evolve and adapt to its surroundings without the need of user input, or knowledge about the algorithm. Both the morphology and controllers were generated automatically using genetic algorithms. Different fitness functions were then used to evaluate these creatures in different scenarios such as: walking, swimming, jumping and following (see fig. 2.1).

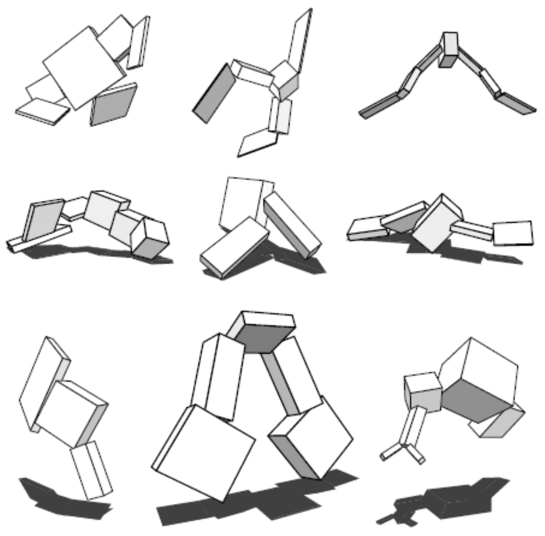

FIGURE 2.1: From top to bottom: Creatures evolved to swim; Creatures evolved to walk; Creatures evolved to jump(extracted from Sims (1994))

2.1.2 Automatic design and Manufacture

Lipson and Pollack (2000) evolved locomotive systems based on simple blocks. These robots were then automatically fabricated using rapid prototyping techniques. Their proposed task was for the robot to walk. The environment used for this task was an infinite horizontal plane. Some of these evolved locomotive systems were symmetric (see fig. 2.2), a possible explanation by the authors is that it is easier for symmetric systems to move in a straight line. Symmetry benefits the system, as it allows the evolved systems to cover a greater area thus gaining more fitness to become the best possible solution. This proposed evolution method is based on the idea that for a system to really adapt and work in the real world, it should be tested in it and not just simulated.

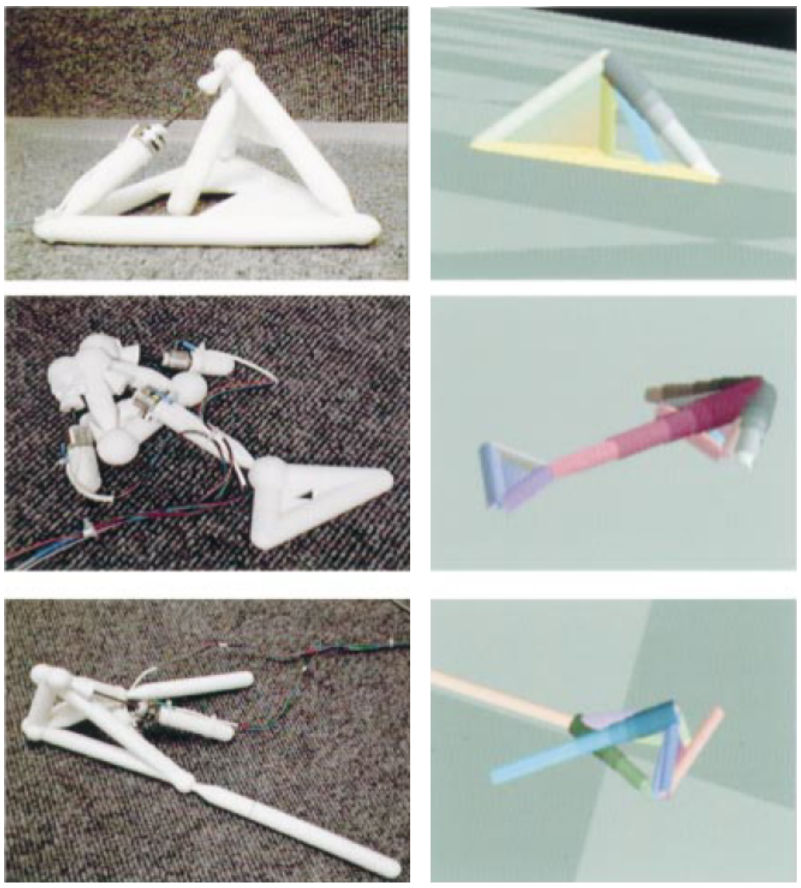

Figure 2.2: The solutions evolved by Lipson and Pollack (right), and their real life prototypes (left).(extracted from (Lipson and Pollack, 2000))

2.2 Genotypes and Phenotypes Generation

Evolutionary algorithms (EA) are algorithms based on biological processes that exist in the real world. These processes include: mutation; selection; reproduction; and recombination.

According to Floreano and Keller (2010), the goal in Evolutionary Robotics is to evolve an artificial body or controller automatically using evolutionary algorithms. Genotypes are commonly represented using a string of bits (0s and 1s), where each gene is represented as a single bit. Genotypes just like in nature contain the information that influences how the phenotype is formed. Genes in a genotype can represent properties, parts, or states of a robot by encoding these properties directly or indirectly. Direct encoding is a straight representation of these properties in the simulation, this means that a gene in the genotype directly maps to the same property in the phenotype. While the indirect encoding specifies how the properties should be generated.

Phenotypes are the realization of the generated genotypes. They represent fully grown agents with the attributes of the previously generated genotypes. These attributes can be a direct representation of the genotype attributes, if direct encoding was used, or they can be the result of the processes applied to the genotype, if indirect encoding was used. These agents as previously mentioned, can be the controller or the morphology of a robot, so a genotype can represent the neural connections of a controller, or the parts that constitute a robot.

To evolve a genotype, the evolutionary algorithm tests it by simulating its behavior to complete the proposed task. How well the genotype performs is determined by the fitness function. The fitness function works by awarding more fitness points the closer the genotype is to completing the task. This genotype is then transformed either directly, or indirectly into a phenotype. To optimize genotypes, there are processes based on Darwins theory of evolution, such as survival of the fittest and blind variations (Gould, 2002), among others. To optimize genotypes we used an approach based on the survival of the fittest.

2.2.1 NeuroEvolution of Augmenting Topologies

Introduced by Stanley and Miikkulainen (2002), NeuroEvolution of Augmenting Topologies (NEAT), is capable of evolving complex solutions and optimizing them. It does so by using the structure as a way to limit the search space of the connection weights. Starting with a minimal structure and then growing it incrementally has been shown to be, not only faster in learning, but also more efficient than minimizing the structure at the end.

NEAT introduces the concept of historical markings. Different genes can be based on the same structure if both derive from the same gene. Knowing their origins, it is easier to choose two genes to mutate or to crossover that are compatible. The historical marking is a global counter, which the authors called the innovation number. When a new gene appears, the counter is incremented and assigned to the new gene.

NEAT also introduces the concept of protection of innovation through speciation. The idea behind this is to, divide the population so that similar topologies compete against themselves instead of the whole population, thus creating groups where each genome will try to be the best solution of that group. To calculate how close or far from a group the genomes are, the authors used the number of excess and disjoint genes from a given pair of genomes. The more disjointed they are, then the less likely it is that they both derived from the same genome.

Calculating the distance δ as a linear combination of the number of disjointed genes D, and the excess E, with an average weight difference of matching genes (includes disabled ones), see 1 .

$$\delta = \frac{C_1 E}{N} + \frac{C_2 D}{N} + C_3 \cdot \overline{W} \tag{1}$$

The coefficients (C_1, C_2, C_3) can be used to adjust the importance of each element (E, D, W). While N represents the number of genes in the bigger genome. With all of this, this method is capable of evolving minimal structures into fully complex ones, while keeping them optimized.

2.3 Grippers

A gripper is a type of an end effector that can be attached to the end of a robotic arm. Its main use is to grasp objects. There are different types of grippers, and while they share the same basic function (to grasp), the different types use different techniques to do so. An impactive gripper applies direct force to grasp the object. A common impactive type of gripper is the claw. The ingressive gripper uses needles to penetrate the surface of the object to grasp, while the astrictive gripper grasps objects by means of suction. The last type of gripper is the contigutive, where the grasping technique is to grasp the object by thermal or chemical methods.

In this study the focus is on the impactive type of gripper. This type of gripper is commonly used with solid objects (mostly dense objects, but also fragile objects).

The first type of grippers was used in an industrial environment, replacing humans in assembly lines to assemble as an example, cars (Devol, 1961). This type of gripper is attached to a robotic arm that is stationary with predefined “states” that make it easier for the gripper to grab the parts to assemble. Nowadays grippers have evolved to be more than just industrialized robots in an assembly line (Tai et al., 2016).

There is a gripper that was created to harvest lettuce (Cho et al., 2002). By using a combination of vision, photoelectric sensors and a controller based on fuzzy logic, it was able to harvest lettuce with a success rate of 94.12% (see fig. 2.3). Grippers capable of performing surgeries have also been created (Bertelsen et al., 2013) (Rateni et al., 2015) . One of the problems associated with grippers that perform surgeries that still remains, is the force feedback (feeling what it is touching and responding accordingly).

While piezoelectric grippers are still difficult to control, they are simple, easy to use and of low power consumption when compared to mechanical grippers, making them a viable alternative to other tools when dealing with micro positioning and handling of small parts (see fig. 2.4).

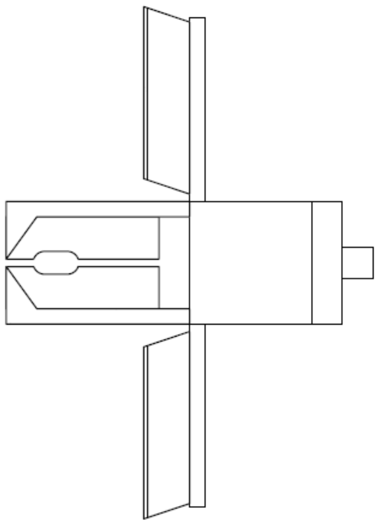

Figure 2.3: The layout of the harvester gripper mentioned above.(extracted from (Cho et al., 2002))

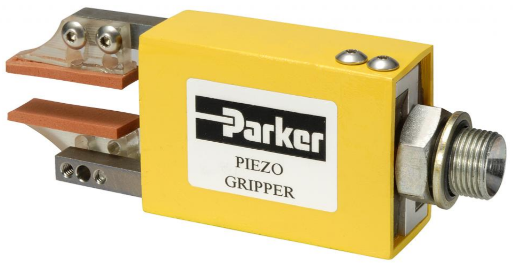

Figure 2.4: An example of a piezoelectric gripper created by Parker Hannifin.(extracted from (McDonnell, 2015))

2.3.1 Haptic Identification of Objects

The work of Homberg et al. (2015) was the creation of a gripper capable of grabbing and identifying objects (see fig. 2.5). The novelty of this solution is the use of proprioceptive sensors (can monitor and control its own internal status). With this type of sensors, it is possible to get the readings of what the gripper is currently grabbing, and to change the internal state accordingly.

The fingers were based on soft manipulators, modified with the addition of a bend sensor to measure the curvature around a certain axis.

By building a relation between objects and the configurations necessary for the gripper to grab them, identifying objects by grabbing them becomes a possibility.

The gripper was trained using the k-means algorithm. Each finger's data was assumed to be independent and was then clustered using k-means to identify objects in a data set.

The authors noted that the only limitation was the resolution of the proprioceptive sensors, so the gripper could distinguish between as many objects as those allowed by the sensors.

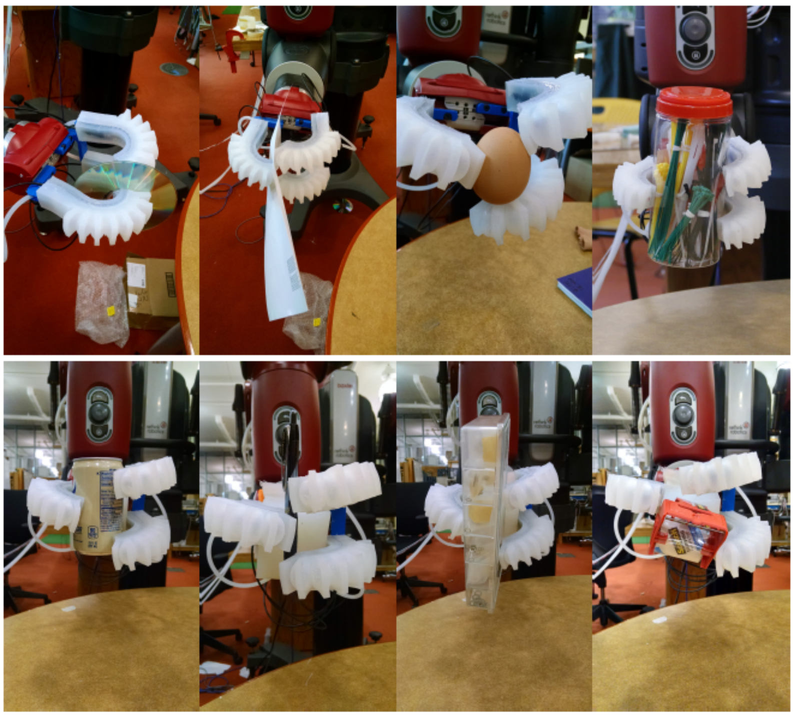
FIGURE 2.5: As we can see, the fingers use a soft material capable of bending to grab objects so the gripper can identify the object it is currently holding.(extracted from Homberg et al. (2015))

2.3.2 Piezoelectric Grippers

Piezoelectric grippers seem to be the future (Monkman, 2000). Due to their low power consumption and ease of use, this type of actuator seems to be a potential candidate to replace other types of actuators such as pneumatic or electromagnetic.

While pneumatic and electromagnetic actuators need to constantly be fed energy in order to maintain the force they are applying, that isn't the case with piezoelectric actuators.

Piezoelectric actuators only use current while they are in motion, but, as long as there is some current passing through even when they are turned off, the same force continues to be applied without any extra energy cost.

The piezoelectric actuator recently has been employed as the gripper itself, eliminating extra energy costs when compared to more traditional actuators.

Although this is a great way to cut costs, the downside is the force applied by the actuator (it is not as strong as a pneumatic based gripper). Piezoelectric actuators, when compared to the traditional actuators, fall short on the force they can apply in a single stroke. However, this is why the piezoelectric gripper is perfect to grasp small components.

2.3.3 Minimally Invasive Surgery with Grippers

Rateni et al. (2015) proposes a novel gripper that uses soft materials to safely interact with biological tissue. The soft material used by this gripper is elastomeric (silicone-based material). The main advantage of this material is that it can get closer to soft tissue without damaging nearby organs.

The gripper itself is composed by three fingers, disposed in a triangular way so that when they close to grasp something they form a pyramid (see fig. 2.6). This way the stability of the gripper during the grabbing of tissue is assured.

The fingers start in an open position; they close as the cables connected to them are pulled. So, when an object is grabbed, the gripper can adjust to its shape.

While results show that it is safe, the authors also claim that manual calibration of the system at the time, is still necessary. This work shows great promise in the use of grippers in Minimally Invasive Surgery.

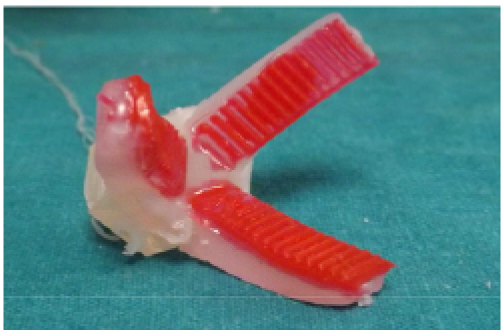

FIGURE 2.6: As we can see, the fingers use a soft material capable of bending to grab objects so the gripper can identify the object it is currently holding.(extracted from Rateni et al. (2015))

Chapter 3

Evolution, Simulation & Prototyping

3.1 Evolution

Evolutionary algorithms are a part of artificial intelligence, they are used to generate viable solutions to a given problem. Viable solutions are solutions that can complete the task and fit within the criteria defined by the user. There are many evolutionary algorithms, with the differences among them being how they represent and search for their solutions. For example, genetic algorithm solutions are usually represented by strings of numbers. By representing solutions as sequences of numbers, it is possible to apply operators such as mutation and crossover in directly on the solution. Another evolutionary algorithm technique is genetic programming. These algorithms' solutions are computer programs usually represented by trees stored in memory, or as strings.

To find the most viable solutions to solve our task, we decided to use the genetic algorithm. Because this algorithm is inspired by processes found in nature that caused morphologies to evolve to what we currently see in our world, and because this type of evolutionary algorithm is used to find optimal solutions, and has been used to evolve morphologies in experiments such as the ones conducted by Sims

(1994), we believe that it may be capable of evolving a viable solution that can grab and lift an object.

The genetic algorithm is based on biological techniques that exist in nature. Evoltuions start with a randomly generated population, consisting of a number of solutions, with each solution being evaluated by how well it performs the task. To evaluate the solution a simulation is created that aims to replicate the real life experiment in a simulated environment. A script is executed during the simulation that allows the solution to complete the task. The closer a member of the population is to completing the task, the better its reward is. This reward is commonly referred to as a fitness value and is calculated during the simulation through a fitness function and is awarded at the end of the simulation.

The fitness function is defined by the user as a means to reward solutions that are closer to completing the task. By rewarding these solutions and by creating the next generations of solutions through mutation or crossing over the different combinations of the highest performing solutions, these new solutions may have a better chance at completing the task.

After all solutions of the population have been evaluated, i.e. a fitness value has been attributed to each solution, the best solutions of that population are chosen to generate a new population. This new population contains the best solutions of the previous generation and new generated solutions based on those with possible mutations. This process is repeated until the last generation is created.

In our experiments, the first generation randomly generates one hundred hands, each starting with three blocks that are positioned randomly in one of five available positions on the palm of the hand (see fig. 3.1). Each block also has its height and length values randomly generated between 2.25cm up to 3.37cm in its height, and from 6.6cm up to 9.9cm. The maximum limit is calculated by multiplying the minimum value (2.25cm and 6.6cm respectively) by 1.5. The width of the block is the only value that remains constant through the evolutionary process, with 1.5cm (see fig. 3.2).

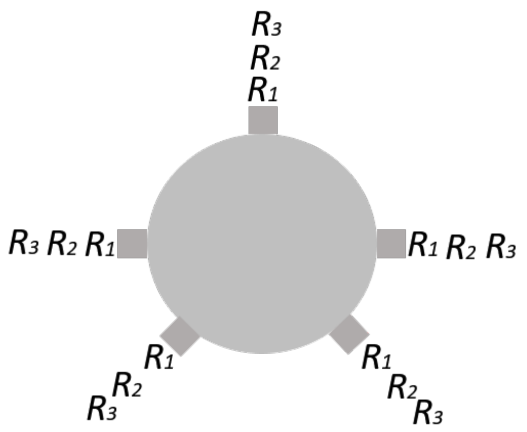

FIGURE 3.1: The first generation of hands is generated with three blocks randomly positioned in the available five positions. R_1, R_2 and R_3 represent the random positions of each block. If a blocks position happens to be shared by another block, the new block is attached to the already existing one.

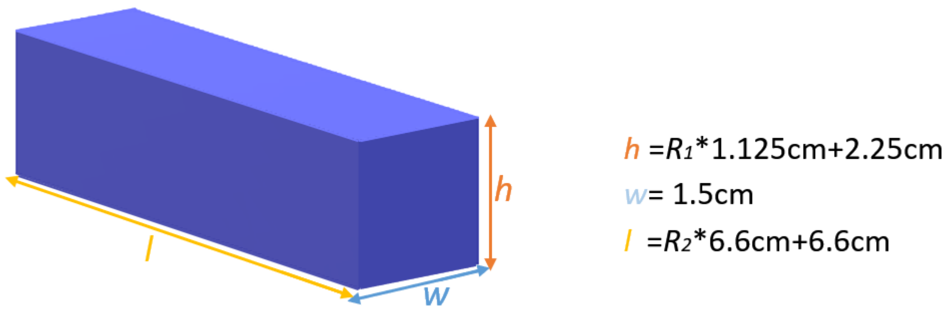

FIGURE 3.2: This figure illustrates the properties of a block. R_1 and R_2 are two randomly generated values ranging from zero to one, that when multiplied by the range value gives us the size to add to the default value.

The task in our evolution is to grab an object and lift that object up and hold it, so we simulated this task by writing a script that the hand would follow to grab and lift a given object. The fitness is calculated at each world step after the object is created. Fitness points are awarded or deducted as shown in 2, with I representing the object's initial position along the Z axis, J representing the object's current position along the Z axis, and P representing the object's previous position along the Z axis:

$$f = \begin{cases} 0, & \text{if the object's current position is outside the hands radius} \\ J - I, & \text{if the object is moving upwards} \\ J - P, & \text{if the object is falling} \\ 0.1, & \text{if the object is not colliding with the ground} \end{cases} \tag{2}$$

The first condition prevents solutions that would push the object out of range to be rewarded with fitness points. Without this condition, the best hands were often the ones that would push the object, instead of grabbing and lifting it. The second and third condition check if the object is being lifted or if it is falling. In the first case, the fitness points awarded are equal to how much the object has been lifted off the ground. In the latter, fitness points are deducted from the current fitness value equal to how much the object has fallen since its previous position.

After all solutions have been evaluated, the ten best are chosen to be cloned to form a new population. During the cloning process, each of the ten best hands is cloned ten times, with each clone having a probability of suffering variations that affect the blocks and their values. These variations can include mutations of the length and height of a block, adding or removing blocks and copying a block structure (a set of blocks connected through joints that share the same position in the palm of the hand that form a "finger") into an available position.

Besides the previously described genetic operators, the evolutionary process has another operator called crossover. A crossover occurs when two members are chosen to reproduce. There are different types of crossover with the simplest being the single point crossover. With single point crossover a point in both parents is chosen at random, the values of both parents are then swapped from that point forward, creating two distinct solutions. The parents' information is stored in arrays/vectors so that it is possible to generate a random index and perform the crossover operation from that index forward (in single point crossover), however our solutions are complex data structures represented by objects and not by numbers and as such we didn't implement this operator.

Each block can suffer mutations or be removed. While for each position available in the palm of the hand a new block can be added or a block structure can be copied (see table. 3.1 for percentages).

This process is repeated until the end of the evolution.

Operations	Frequency	Probability (%)
Mutation	For each block	10
Addition	For each position	10
Removal	For each block	4
Copy a block structure	For each position	4

Table 3.1: Each operation with their frequency and probability.

3.2 Simulator

To run the described experiments, a simulator was built using Java with the support of an open-source library and physics engine called Open Dynamics Engine (ODE) created by Smith (2000). This library was helpful in creating a simulated environment where objects in it are affected by forces in the same way they would be if these experiments were done in the real world.

This simulated world consists of the following objects: A support, a hand and its generated blocks, and the object to grab and lift (see fig. 3.3).

The support is a static object created above the hand and can be thought of as the robot "arm" that connects to the hand in the real world. This support is necessary since, ODE is a physics engine and as such, when an object is created inside the simulation, it is immediately affected by the forces acting in the simulation. This means that, as soon as the hand is created in the simulation, it would fall to the ground due to the effect of gravity, and considering the purpose of this work is to generate a hand capable of functioning in the real world, we could not ignore the effects of gravity in our simulation, so the solution was to create a static object which we called “support”.

The static object is not affected by any of the forces present in the simulation, and its only purpose is to be attached to the hand by means of a slider joint. With this object, we created a mechanism that allows us to control when and how fast the hand should come back up.

The object to grab and lift is a main part of these experiments. This object can take any primitive solid form that exists in ODE, or that can be imported from a 3D object (by loading a custom triangle mesh). The primitive objects already supported by ODE are: rectangles, capsules, cylinders, and spheres.

The primitive objects have parameters such as radius and lengths that can be adapted to better fit our experiments. All the objects have density and mass, and while ODE calculates both of them, taking into account the lengths of the object and its radius (if necessary), we found it necessary to readjust the mass to one similar to the one in aluminum cans as it is a common object found in the real world that can be replicated using primitive objects.

The generated blocks that compose the "fingers" that belong to the hand are generated by the evolutionary process. They can be added, removed, mutated or replicated during evolution. The minimum values for the height and length of a block were defined so that there was enough space to fit a servo for control. Only the width of the blocks was fixed and not subject to mutations by the evolutionary process.

The width is a constant due to initial experiments where we observed that the width of the block was interfering with how the hand could grab the object, being detrimental to solutions as these solutions could not completely close to grasp the object. The length and height can be mutated up to 1.5 times their default size (6.6cm for the length and 2.25cm for the height property).

Block dimensions were calculated to leave enough space for a servo to fit in. The servo we used to replicate these dimensions was the HXT900 Micro Servo. This servo is small but can produce enough force to push objects up to 1.6Kg.

The palm of the hand is a cylinder with 10 centimeters of diameter. In initial experiments, it had twelve positions where blocks could potentially connect, this was later changed to five positions because the blocks were colliding with other blocks in neighboring positions, causing strange behaviors in the simulation, making it unstable. Table 3.2 presents a summary of the objects used during our experiments as well as their properties.

Object	Mass (Kg)	Height (cm)	Width (cm)	Length (cm)
Object	0.0152	User Defined	User Defined	User Defined
Block	0.014	2.25 up to 3.37	1.5	6.6 up to 13.2
Hand	0.059	N/A	5	0.5

Table 3.2: Values for each object used in the simulation. The object's mass is based on the average weight of an aluminum can. The block and hand mass is based on the values generated by the 3D printing software.

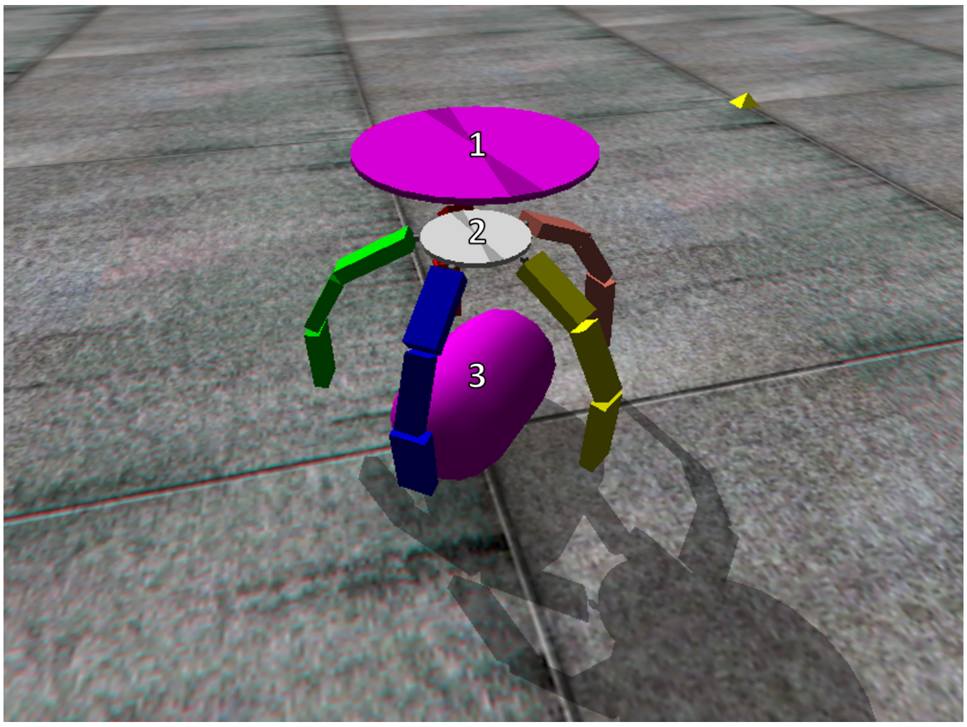

Figure 3.3: 1)the support connecting to the palm of the hand; 2)The palm of the hand connecting to multiple blocks; 3)A capsule as the object this solution is going to try to grab and lift.

ODE does not have a defined unit system. This means that all the values present in the simulation are free for interpretation until we assign them a unit. For example, we can create a cube that is 3x3x3, and ODE won't assign any units to this cube. It just creates it, so we have to assign some unit of measurement to it.

Depending on the scope of the problem, this unit can be centimeters, meters, or any other unit of measurement that fits the experiments.

This kind of versatility can be good or bad. It's good because it allows us to simulate big structures without needing to represent them one to one, which means we don't need a very powerful machine to run experiments.

Other parameters that affect the simulation (see table. 3.3):

World step controls the amount of time, in seconds, passed between each step. How much a step is worth is assigned by the user. ODE recommends to have a small step size so that the simulation runs smoothly. A bigger time step leads to a bigger jump between steps, which can lead to unstable simulations. In this case we are using steps that move time forward 0.0009 seconds.

Error Reduction Parameter (ERP) is used at each time step to align bodies that are connected by a joint. This is necessary because, as the simulation runs bodies tend to drift away from their positions. If an ERP value is set, a force is applied at the joint to force the bodies back together. This value is user defined, and has a range from zero (no error is corrected) to one (tries to correct all the errors), however the latter is impossible due to internal approximations. And because correcting joint errors is a demanding task, it is not recommended to set the value to one. Correcting all errors can lead to slow simulations, especially if there are a lot of objects connected by joints.

Constant Force Mixing (CFM) combined with ERP allows us to simulate different types of materials by changing their values. CFM, is what determines if a constraint is "soft" or "hard". In an hard constraint, there are rules that can never be broken, for example, objects can not by default penetrate the ground's surface, this is called a hard constraint. While a soft constraint allows objects to penetrate, or clip through others. This type of constraint can be useful to simulate various types of soft materials.

Damping is used to reduce instability. It reduces both the angular and linear velocity until both values are below threshold values.

Simulation Parameter	Values
World Step	0.0009 sec.
Error Reduction Parameter	0.8 (Max Value)
Joint	When connected to the palm of the hand:
	- -45° to 90° of rotation;
	When connected to another block:
	- 30° to 0° of rotation;
Constant Force Mixing	0.00001 (Default Value)
Damping	Angular Scale: 0.00673
	Linear Scale: 0.01832

Table 3.3: Values for each simulation parameter being used during the simulation. To prevent blocks from going backwards when opening the hand, joints between blocks can not rotate more than their upper limit value (0°). The damping scale values were taken from examples provided by ODE.

To prevent clipping between the object and the blocks and the palm of the hand, the simulation starts by first opening the hand. This is done by rotating each blocks' joint angle close to the joints maximum opening angle (45° degrees for the first blocks and 0° for the rest).

After fully opening the hand, the object appears, thus preventing any possible clipping between the object and the blocks or the palm of the hand.

To check if the blocks are touching or grabbing the object, we had to build a collision detection system based on the one provided by the ODE demo package. For each contact the ODE collide function is called which generates the contact information of two given objects geom (body) that could be intersecting. Then, for each contact it checks if one of the geoms in the contact belongs to the object and if so, it then proceeds to check the other geom to see if it belongs to a block connected to the palm of the hand. If both conditions are met, then that means the object is colliding with a block connected to the palm of the hand, therefore it is being grabbed.

The system starts by creating a buffer of contacts with a limit of contacts, by default it's 64 contact joints.

Contacts are the objects' surfaces, and contact joints are created by the collision detection system and are deleted as soon as the current step ends. They serve to prevent objects from clipping into each other by using an "outgoing" velocity. Although there is a default limit of 64 contact joints, we ran experiments with 32 and 128 contact joints. The results showed that the simulation is faster with 32 contact joints while maintaining the same fitness values as a simulation with 64 contact joints when simulating with a primitive object such as a capsule, but returns different values when simulating with complex or imported objects. While simulations with 128 contact joints took longer to finish and had different fitness values even for primitive objects.

Each contact has surface parameters that can be set. These parameters allow us to change how bouncy or soft the surface is, or how slippery, amongst other properties, an object is. See table 3.4 for the values corresponding to those parameters.

Surfaces	Values
Mode	dContactSlip1
	dContactSlip2
	dContactApprox11
	dContactApprox12
	dContactSofCFM
softcfm	10^{-5}
mu	dInfinity
slip1 and slip2	150

Table 3.4: Values for each surface parameter of a contact. Some parameters such as softcfm and slip1 and slip2 need to have their corresponding flags set before being used.

3.3 Prototyping

Processing is a Java software and library developed by the Processing Foundation (Fry and Reas, 2001) with the purpose of easing the creation of software that can create/generate art. Processing is used all over the world in different ways, from computer courses in art schools to exhibitions in museums such as the Exploratorium in San Francisco or the Museum of Modern Arts in New York. It also has a very prominent place in research, being used to visualize data in Nokia, Yahoo! and General Motors.

In this work, Processing was used to assemble the generated blocks from the main evaluation process with pre-made parts to create the final piece that could be 3D printed and used in a real world experiments.

In Processing, the block that looks like a fin, and is used to connect the block to another block or to the hand, and the block with the space to insert the servo (fig. 3.4), are combined with the generated block (fig. 3.5). To do this, the block that looks like a fin is inserted at one of the ends of the generated block, and the block with the space for the servo is inserted at the other end.

Because the generated block can have a different width than that of the fin block, a width offset was calculated. With this offset, we can now center the fin block. The same process is repeated with the servo block, but at the other end. The newly assembled piece (fig. 3.6) is then exported into a format that can be read by the 3D printing software. This process is repeated for each generated block in a hand.

Figure 3.7 represents a 3D model of the complete hand that was generated.

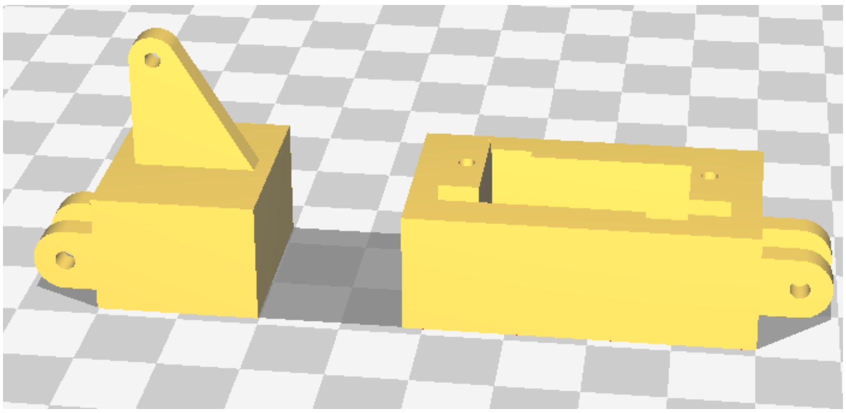

FIGURE 3.4: The fin and servo pieces created outside the simulation.

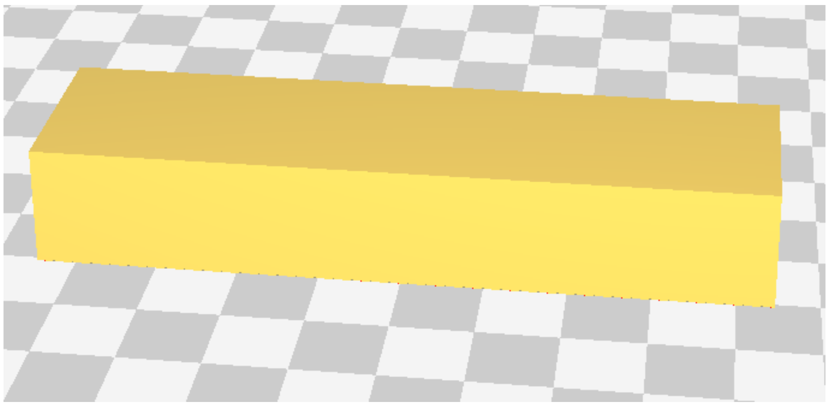

FIGURE 3.5: Example of a generated block from the simulation.

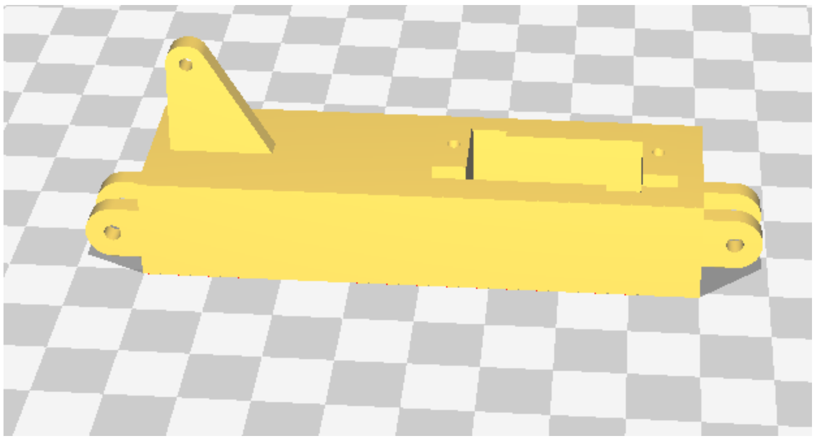

FIGURE 3.6: The final block with both the servo and fin combined with the generated block.

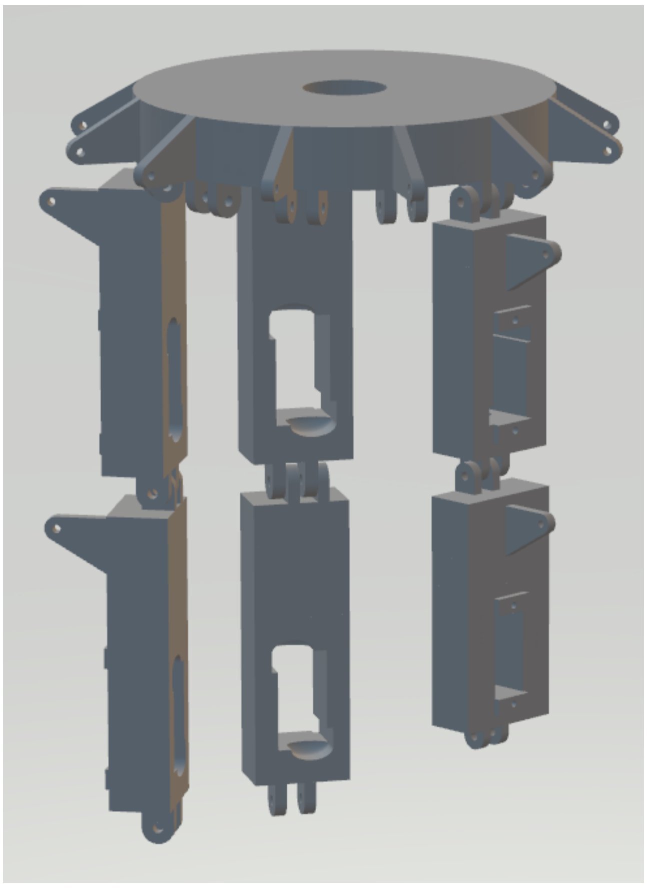

FIGURE 3.7: The gripper assembled with the modified block pieces. This 3D model represents the gripper shown in 3.7.

Chapter 4

Results

The goal of these experiments was to evolve a gripper's morphology that could grab and lift specific objects through genetic algorithms. A hand that would be capable of grabbing, lifting and holding objects independent of their rotation and size. In our experiments, the initial setup consisted in a hand composed by three randomly generated blocks in three randomly chosen positions of the five available in the palm of the hand, and an object that could be a capsule, a rectangle, a cylinder or a sphere.

In these experiments, the hand followed a script with three steps (see appendix E.1 for an example of a successful hand, and appendix F.1 for an unsuccessful hand):

- Open hand: Each block has a hinge attached to it, and at this step the block rotates over the hinge's axis towards the outside, therefore "opening" the hand;
- Close hand: After the blocks reach their maximum rotation over the hinge outwards, they start to rotate inwards over the hinge's axis, as if "closing" the hand;
- Move up: As soon as the blocks collide with the object, the hand starts moving up;

To simulate the opening and closing of the hand, a torque of 0.156 N.m is applied in the joints connecting the blocks to the hand. This torque value is divided by the number of parents a block that is not connected to the hand has.

Block morphology was generated with an evolutionary algorithm, more specifically a genetic algorithm. Each generation comprised of 100 genomes, each encoding the morphology of a hand. The hands' morphology properties values were stored in variables in a Java class. Each hand evaluation consisted in four samples, except for hands evaluated to grab a sphere due to its symmetry. In each sample the initial position of the object has a different angle along the x axis (see fig. 4.1). The first sample had the object at 0^o of rotation, and the following samples rotated the object in increments of 45^o. The fitness value attributed to a hand was the average fitness value of the four samples. Each sample had a duration of 4000 steps, with each step moving the simulation forward 0.0009 seconds.

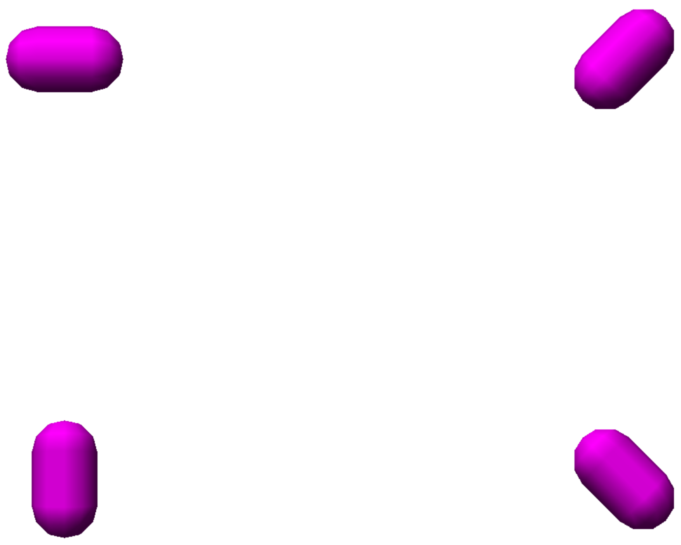

FIGURE 4.1: For evaluations that used four samples, each sample corresponds to one of these angles demonstrated with a capsule object. Starting with 0^o rotation at the top left corner; 45^o rotation at the top right; 90^o rotation at the bottom left; 135^o rotation at the bottom right;

The top ten hands are then chosen to populate the next generation using an elitist approach, only the top ten of the evaluated generation are selected with the rest of the hands being discarded.

Each of those top ten hands is cloned ten times, with each clone having a 10% chance of mutating for each block, changing its length or height. For each block there is also a 4% chance of it being removed from the hand. For each position there is a 10% chance of adding a new block and a 4% chance of replicating an already existing block structure, this means that it can copy already existing chains of blocks from one position to other positions.

To evaluate solutions, we created a fitness function that awards fitness points to our solutions. A fitness function, is a type of function that allows us to evaluate how close a solution is to reaching its goal. According to Floreano and Urzelai (2000), a fitness function can be defined in three axes:

- **Functional or behavioural:** A functional fitness function is based on measurements of the components involved in generating a behaviour, such as, the velocity with which the gripper opens or closes. A behavioural fitness function however, is based on the effects of said behaviour, such as, closing the gripper too fast can lead to objects to slip from its grasp;
- **External or internal:** a fitness function is considered external if its calculations are based on variables not readily available to the robot and only available to an external observer. While an internal fitness function is based on the robot having the variables available at the moment through sensors for example;
- **Explicit or implicit:** An explicit or implicit fitness function depends on the amount and type of constraints explicitly imposed on solutions, such as, specify the size of a blocks length, height or width;

A fitness function works by awarding fitness points to a solution during the simulation. Fitness points are awarded as the solution meets conditions created by the user. The amount of fitness points is also defined by the user and it varies depending on the task.

Fitness points are rewarded by the height reached by the object, this means that the higher the object is, the higher the fitness points awarded to that hand. Fitness points are deducted in the same way as they are awarded, which means, the bigger the fall, the more points are deducted from the final fitness value. Fitness points are also awarded if the object is not touching the ground. This condition is met if the object's geom is not colliding with the plane's (ground) geom.

Fitness points are awarded according to 3 as soon as the "close hand" step of the script starts, and are accumulated during the simulation:

$$f = \begin{cases} 0, & \text{if the object's current position is outside the hands radius} \\ J - I, & \text{if the object is moving upwards} \\ J - P, & \text{if the object is falling} \\ 0.1, & \text{if the object is not colliding with the ground} \end{cases} \tag{3}$$

Where I is the starting position for the object along the Z axis, J is the object's current position along the Z axis, and P corresponds to the position along the Z axis immediately before the current position.

To reduce the number of false positive results, the fitness function used was adjusted to benefit solutions where the object being picked up does not cross a threshold, that threshold being the hands radius. With this condition, solutions where the blocks would flick the object far away, therefore being awarded fitness points, are discarded.

Before this condition was added to the fitness function, most solutions would evolve hands that would kick the object across great distances, gaining altitude due to the force applied in the kick, leading to incorrect solutions because the object was never grabbed by the hand.

4.1 General Results

Evaluations were conducted using ten different seed values. An evaluation was created for each size (large, medium, small) of each primitive object in ODE (capsule, cylinder, rectangle and sphere), giving us a total of 120 hands across all evaluations.

Although these 120 solutions are valid, as the hand grabs the object as it should, most of them do not perform as well as they should have to be considered an actual valid solution. To be considered an actual valid solution, a hand should be able to grab, lift and then hold the object until the simulation terminates. As such, we need to filter these solutions that were evaluated and make sure they can actually grab the object without letting it fall. This reduces the number of solutions that can be considered actual valid solutions to 28% of the total number of solutions.

We examined each of the previously evaluated hands in each sample that was used during the evaluation process. If the hand was evaluated with an object that had four samples, each sample contributes a maximum of 25% to the total fitness value. So during this examination if the evaluated hand grabs and lifts the object in a given sample, we award that sample its maximum percentage value. If the hand is able to grab the object but drops the object, we award it with half of the maximum value, which is 12.5%. If the hand is unable to grab and lift the object, we dont award it any fitness.

Hands evaluated to grab and lift sphere objects are evaluated using one sample, so the maximum contribution of that sample to the total fitness value is 100%. In this case, we award a hand that successfully grabs and lifts the object its maximum percentage value, 100%. If the hand grabs the object but the object eventually falls, we award it half of its maximum percentage value, 50%. And 0% if the hand can't grab the object at all.

After filtering the 120 hands, we can see that 28% of the evaluated hands can in fact grab the object in all samples without dropping it. Hands that could not grab the object and lift it at all make up for 33% of the results, and hands that could grab and lift the object in at least two of the four samples, correspond to 39% (see fig. 4.2).

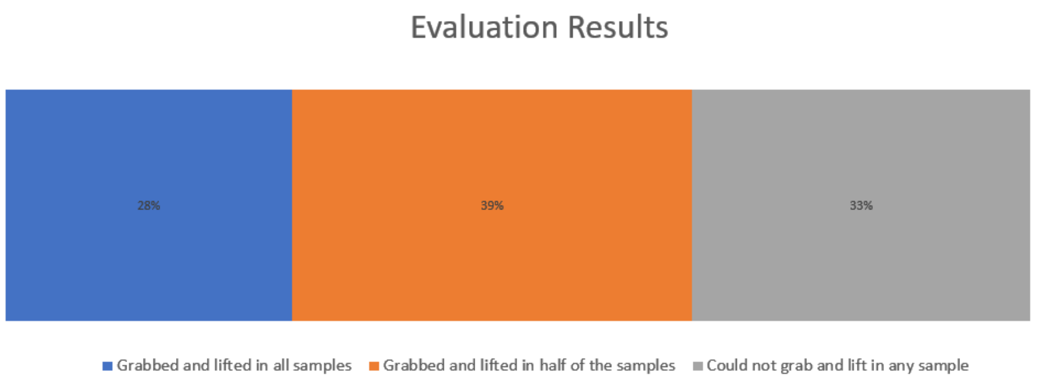

FIGURE 4.2: Evaluations results breakdown.

Of those 28% that can grab and lift the object in all samples, 82% of them were hands evolved to grab the sphere object. Hands evolved to grab capsules represent 15% of these results, with the final 3% corresponding to hands evolved to grab rectangles. None of the hands evolved to grab cylinders were able to grab the object in all samples without dropping it (see fig. 4.3).

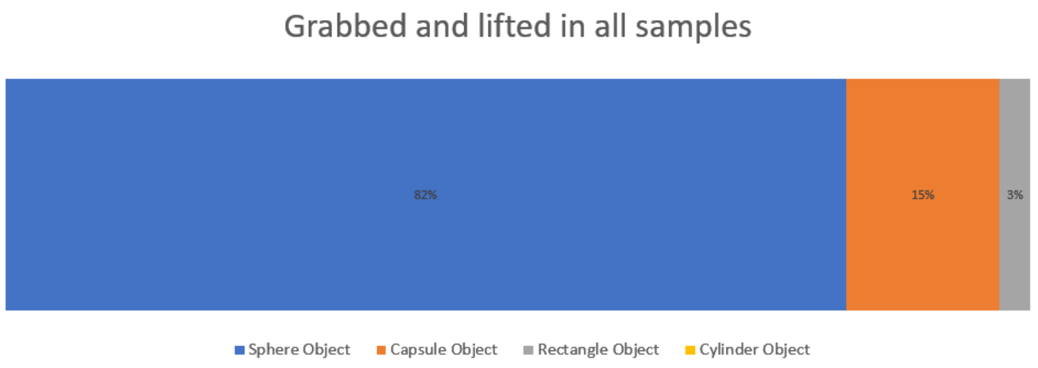

FIGURE 4.3: Breakdown of hands that grabbed the object in all samples.

Hands that failed to grab and lift the object the most, were hands evolved to grab rectangles (45%). This can be explained by the rectangle's hard edges, unlike those of the capsule, cylinder and sphere, making it harder for the hands to wrap around it.

Close behind as one of the hardest objects to grab and lift, is the cylinder (43%). Although it doesn't have the same hard edges as the rectangle, the cylinder unlike the capsule and the sphere has flat surfaces, which makes it harder to grab the object in some samples. Hands evolved to grab capsules make the final 12% in this category (see fig. 4.4). Hands evolved to grab spheres never failed to grab the object and lift it of the ground.

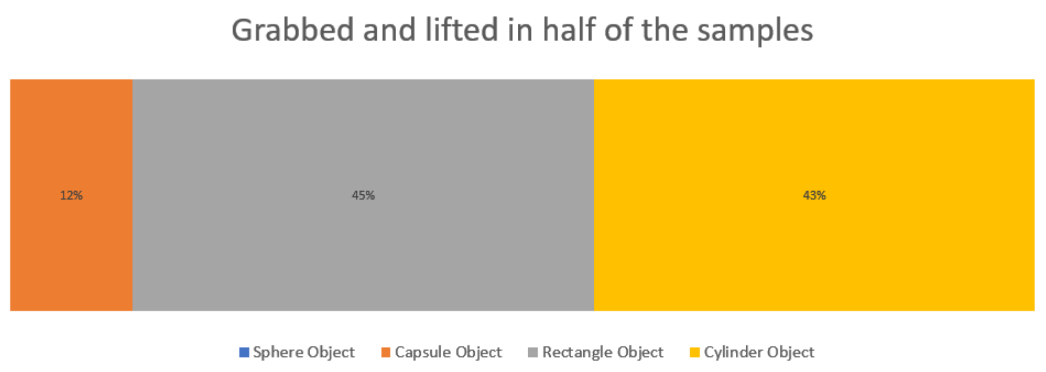

FIGURE 4.4: Breakdown of hands that grabbed the object in half of the samples.

The largest group is the one with hands that grab and lift the object in at least half the samples. Most evolved hands end up in this group because the third sample is in a completely different orientation from the starting angle. If we look at the object from above, we can say that the starting angle with 0° of rotation is the object facing a vertical direction, while the third sample's angle has the object facing an horizontal direction, this makes it extremely difficult for hands to evolve to grab all four different samples.

And so, many of the hands evaluated with four samples, evolved into successfully grabbing two, sometimes three of the four proposed samples.

A large percentage of hands in this group are those evolved to grab capsules (43%), with hands evolved to grab cylinders and rectangles sharing similar percentages, 28% and 24% respectively. There were two hands that evolved to grab spheres that let the object fall once they reached the limit, this happened because in both solutions one of the hands' sides was left without any blocks to hold the sphere.

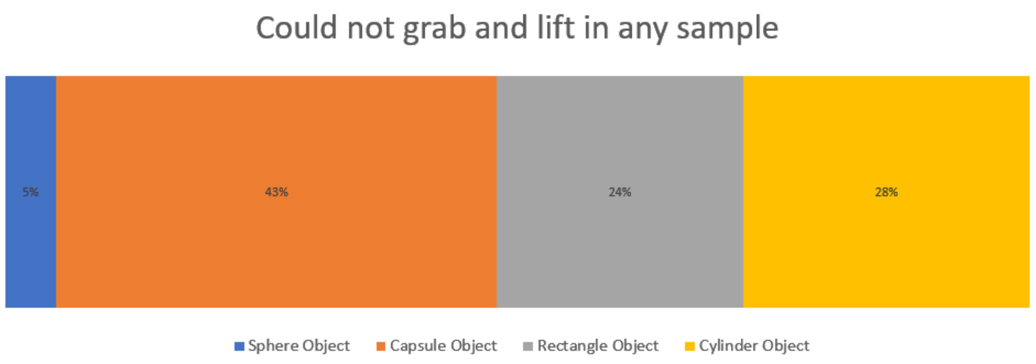

FIGURE 4.5: Breakdown of hands that grabbed the object less than half of the samples.

When dealing with bigger objects, the evolutionary process favored hands with more blocks, while hands dealing with medium and smaller objects evolved to use fewer blocks (see fig. 4.6), sometimes even less than the initial number of three blocks. The only exception to this are hands evolved to grab spheres, due to their round configuration, all hands evolved to have no fewer than five blocks across all object sizes.

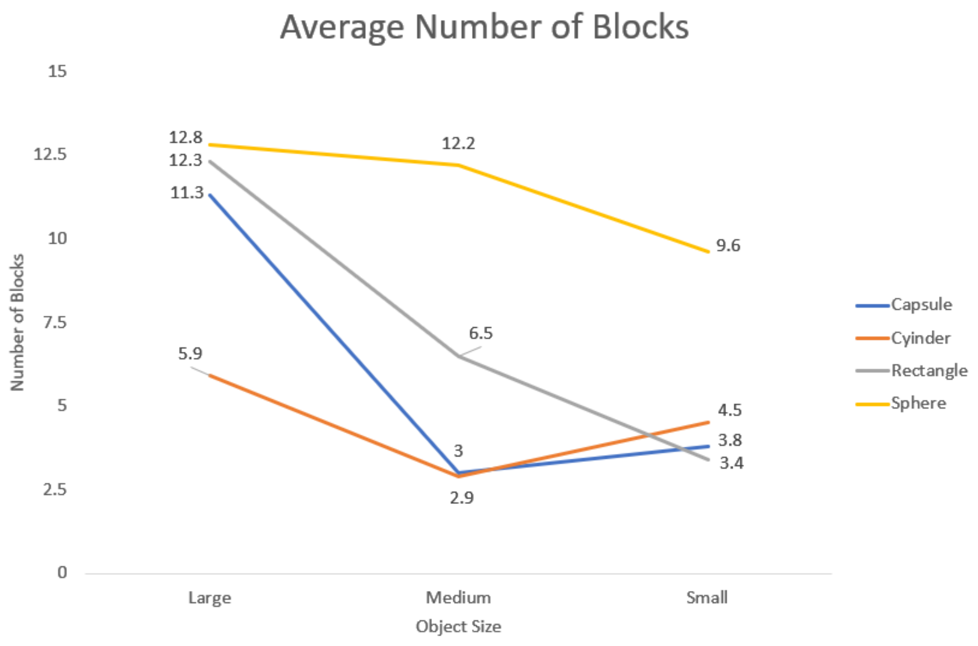

FIGURE 4.6: Average number of blocks by object size.

Medium sized objects seemed to be the hardest to grab and lift, as no hand with the exception of those evolved to grab medium spheres, managed to grab and lift medium objects in all samples.

Medium objects are smaller than large objects, so the torque necessary to grab and lift said objects is also smaller. One way to apply less torque, is to use less blocks, and with less blocks there are fewer chances of a clone to suffer a mutation, however it also means that the mutations that happen will fall on blocks that suffered mutations already.

This also happens to hands evolved to grab smaller objects, but due to their smaller size, these objects are normally grabbed by blocks squeezing the object between them.

The medium object not being as large as the large object, leads to solutions with fewer blocks, combined with not being able to squeeze the object between the blocks like smaller hands, leads to a worse performance by hands trying to grab medium objects.

There is also a tendency across all seeds for the best hands to use the same number of blocks to grab the same object. Hands with more blocks than the average number of blocks for best hands performed worse. What the best hands have in common is that, at least 50% of the positions used are shared across all seeds, this means that at least half of the blocks of the best hands across all seeds for the same object, share the same positions.

Knowing this, the evolutionary process can be sped up, and some previous local maxima can potentially be overcome, as we can now define starting positions and a minimum number of blocks from the for a hand to start with for a given object.

We can also conclude that a completely different orientation is much harder to grab (see the first angle of 0^{o} with the third of 90^{o}), and because the other two angles are much closer to that first angle, it is understandable why most hands' evolutions have a worse success rate when it comes to grabbing and lifting the object in its third angle.

4.2 Capsule Results

Hands evolved that could grab and lift large capsules in all four samples always used fifteen blocks (fig. 4.7), while hands evolved to grab medium capsules mostly used just three blocks (fig. 4.8), with varying degrees of success, but none of them could grab and lift the object in all four samples.

Only one hand evolved to grab and lift small capsules managed to do it in all samples (see fig. 4.9).

The fitness graph for the best evolved morphology at each generation and the average values of all 10 seeds that were evolved to grab and lift large, medium and small capsules can be seen in Appendices A.1, A.2 and A.3 respectively.

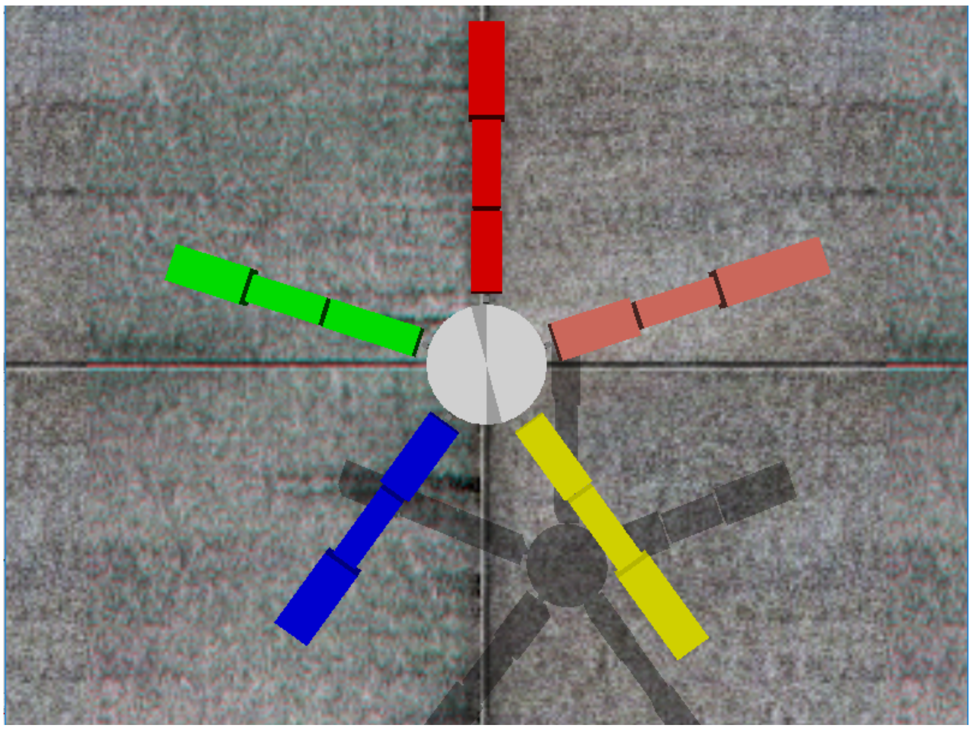
FIGURE 4.7: Best hand to grab and lift large capsules.

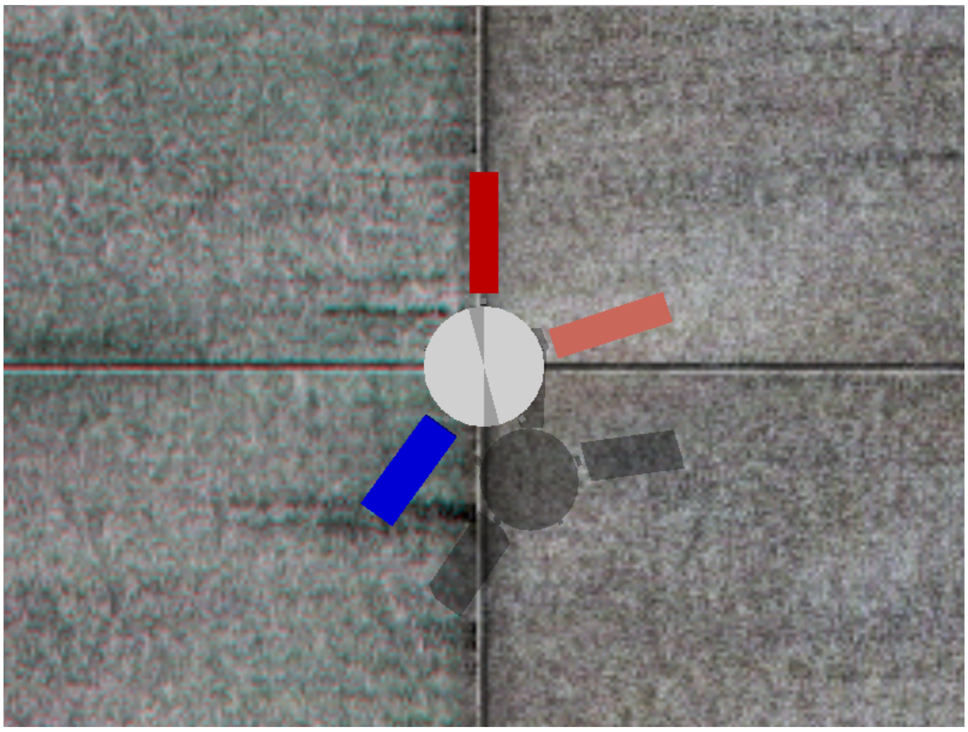
FIGURE 4.8: Best hand to grab and lift medium capsules.

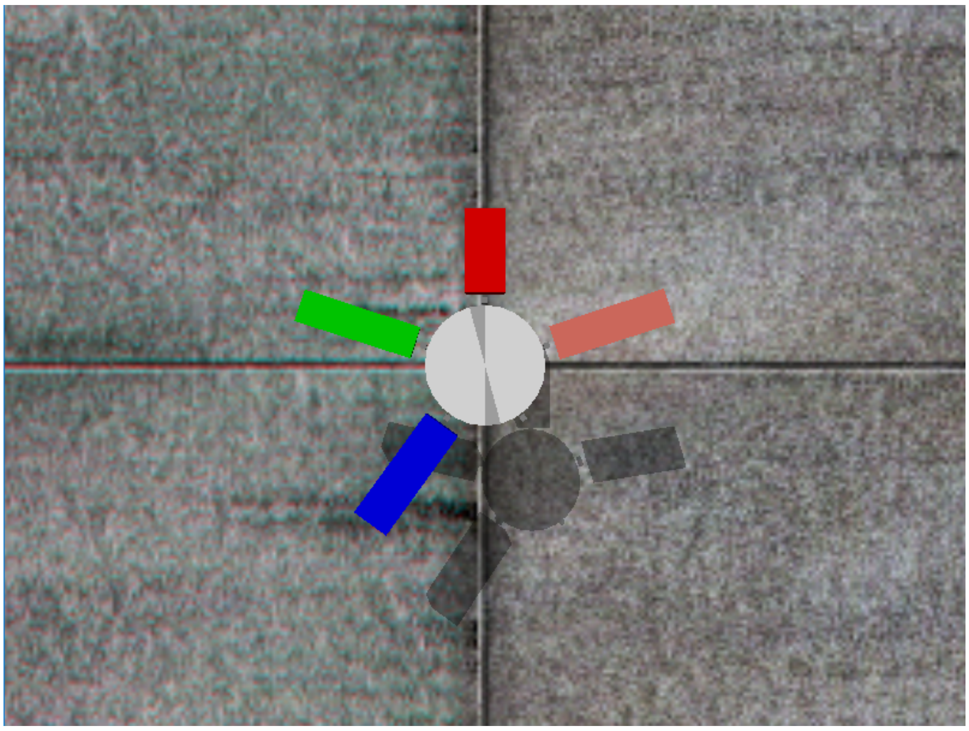
FIGURE 4.9: Best hand to grab and lift small capsules.

4.3 Cylinder Results

To grab large cylinders all hands across the ten seeds evolved to have six blocks split in pairs, with some variation of one pair in a position opposite to the other two pairs (fig. 4.10). Although every hand evolved used just six blocks, all the hands are unable to grab the object, or drops it in at least one of the four samples.

The medium cylinder hands that could grab and lift evolved to have at most three blocks (see fig. 4.11), with all solutions failing to grab and lift the object across all four samples, lifting and grabbing it at most in two of the four samples.

Only one in ten hands was able to grab and lift the small cylinder in one sample (see fig. 4.12), the rest of the solutions failed to even grab the small cylinder.

The fitness graph for the best evolved morphology at each generation and the average values of all 10 seeds that were evolved to grab and lift large, medium and small cylinders can be seen in Appendices B.1, B.2 and B.3 respectively.

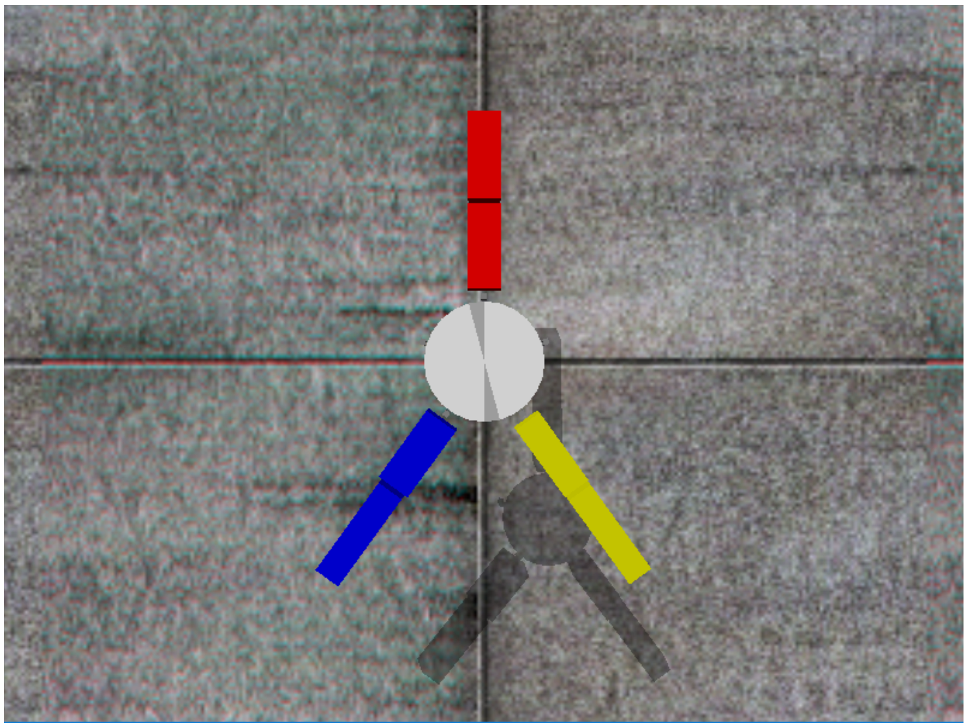

FIGURE 4.10: Best hand to grab and lift large cylinders.

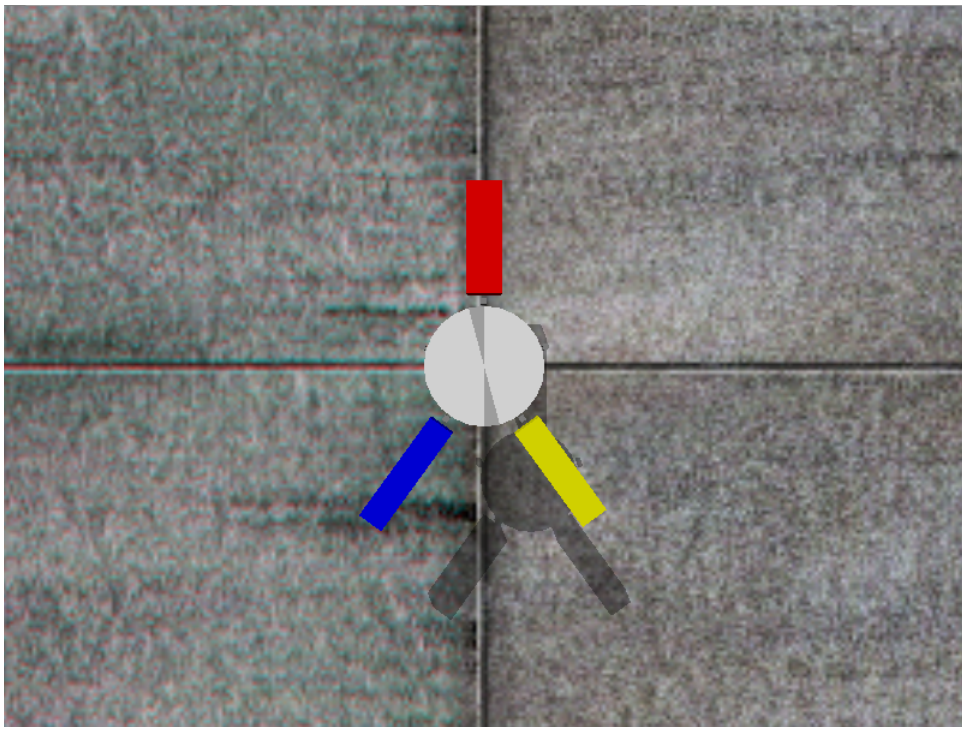

FIGURE 4.11: Best hand to grab and lift medium cylinders.

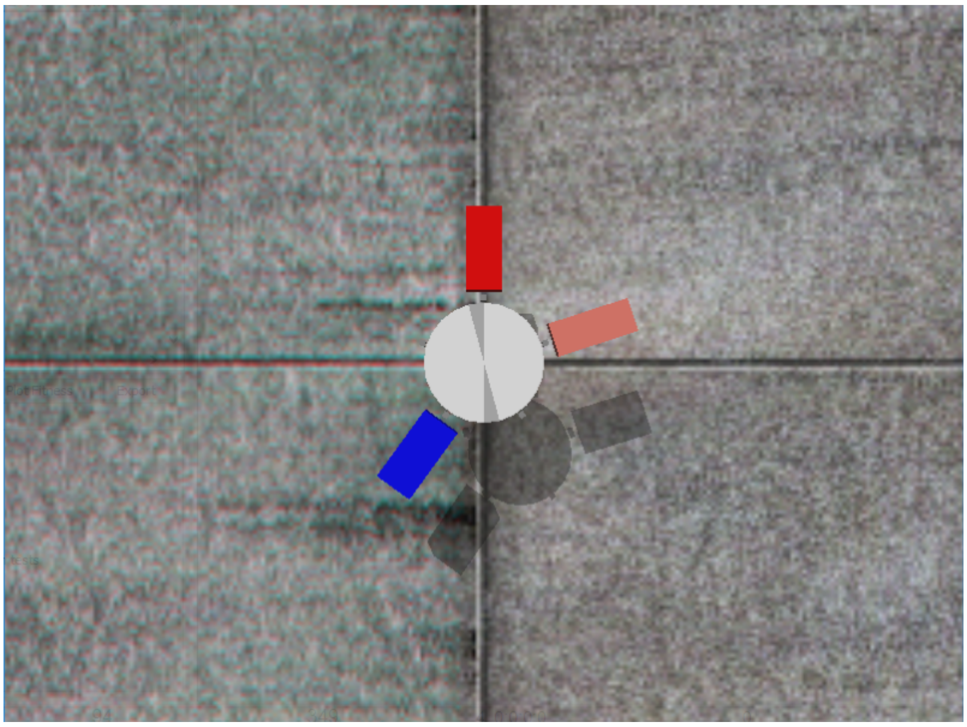

FIGURE 4.12: Best hand to grab and lift small cylinders.

4.4 Rectangle Results

Most of the evaluated hands evolved to use twelve blocks to grab the large rectangle, but only one of them could grab and lift the object in all four samples (see fig. 4.13). Solutions with more than twelve blocks had a worse fitness value.

Hands that used ten blocks to grab medium rectangles, could grab it in all four samples, however they could not hold it until the end of the simulation (see fig. 4.14).

Although the best solution to grab small rectangles only uses two blocks, it could still only grab the rectangle in two of the four samples (see fig. 4.15).

The fitness graph for the best evolved morphology at each generation and the average values of all 10 seeds that were evolved to grab and lift large, medium and small capsules can be seen in Appendices C.1, C.2 and C.3 respectively.

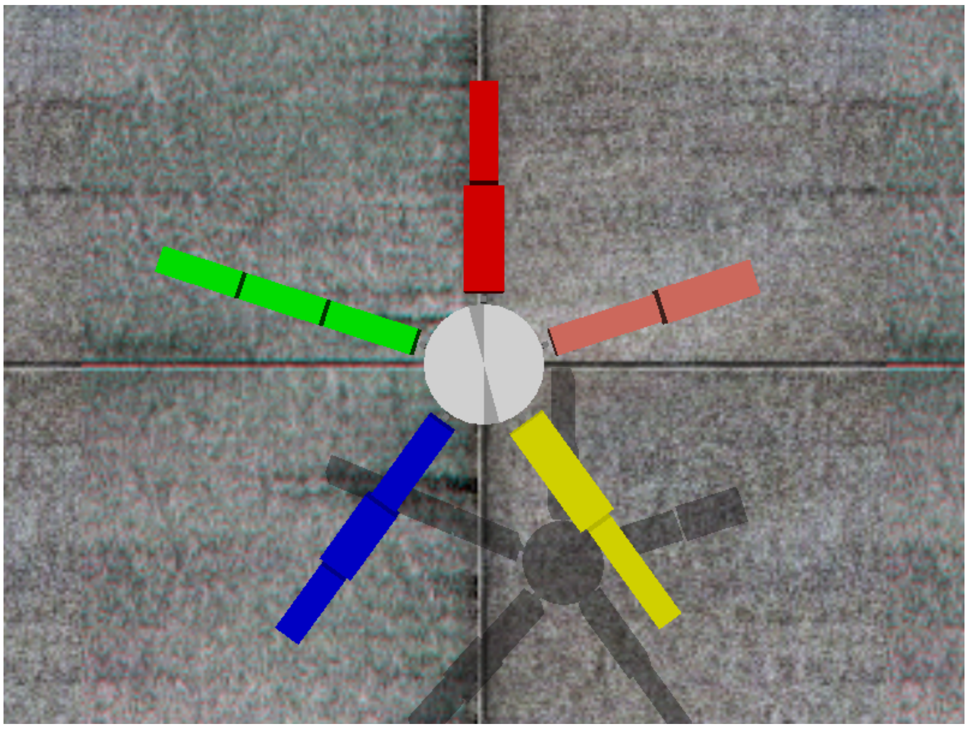

FIGURE 4.13: Best hand to grab and lift large rectangles.

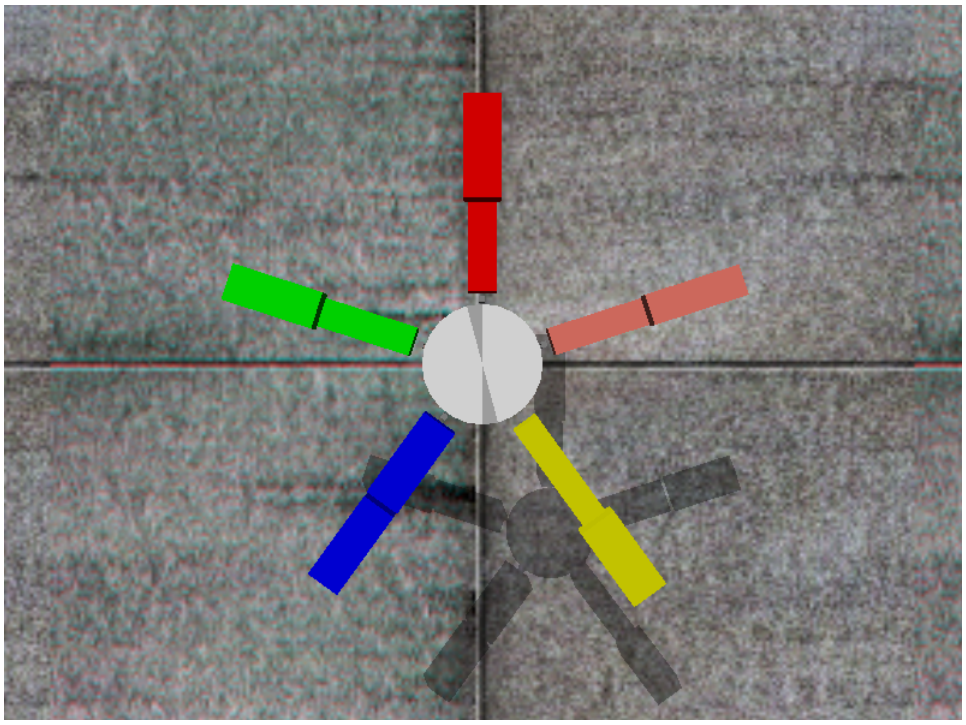

FIGURE 4.14: Best hand to grab and lift medium rectangles.

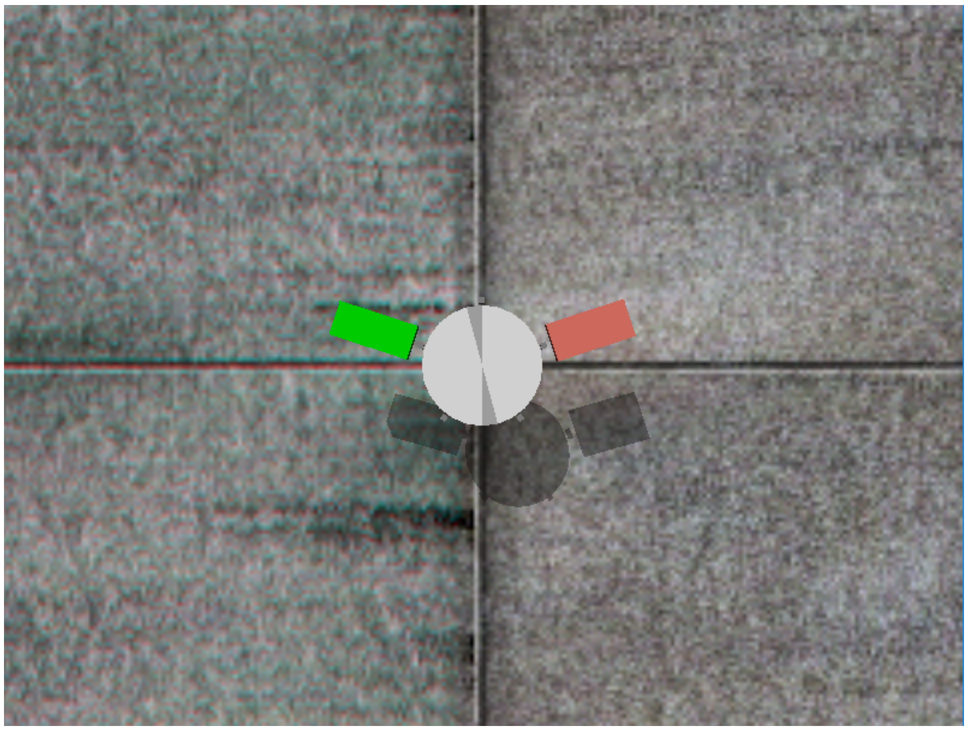

FIGURE 4.15: Best hand to grab and lift small rectangles.

4.5 Sphere Results

The best hands to grab large spheres are the ones that evolved to have fifteen blocks (fig. 4.16), while twelve blocks seems to be the optimal solution to grab medium spheres (fig. 4.17). Hands with more blocks had a lower fitness value than hands that had fewer blocks, with eight blocks being the best configuration to grab small spheres (see fig. 4.18).

The fitness graph for the best evolved morphology at each generation and the average values of all 10 seeds that were evolved to grab and lift large, medium and small capsules can be seen in Appendices D.1, D.2 and D.3 respectively.

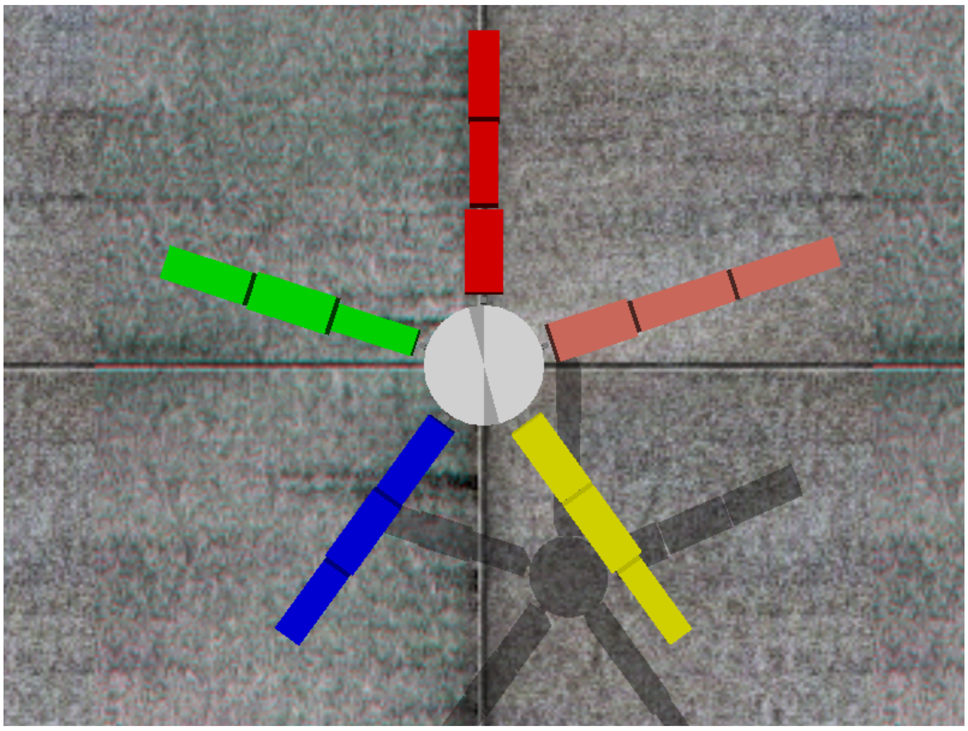

FIGURE 4.16: Best hand to grab and lift large spheres.

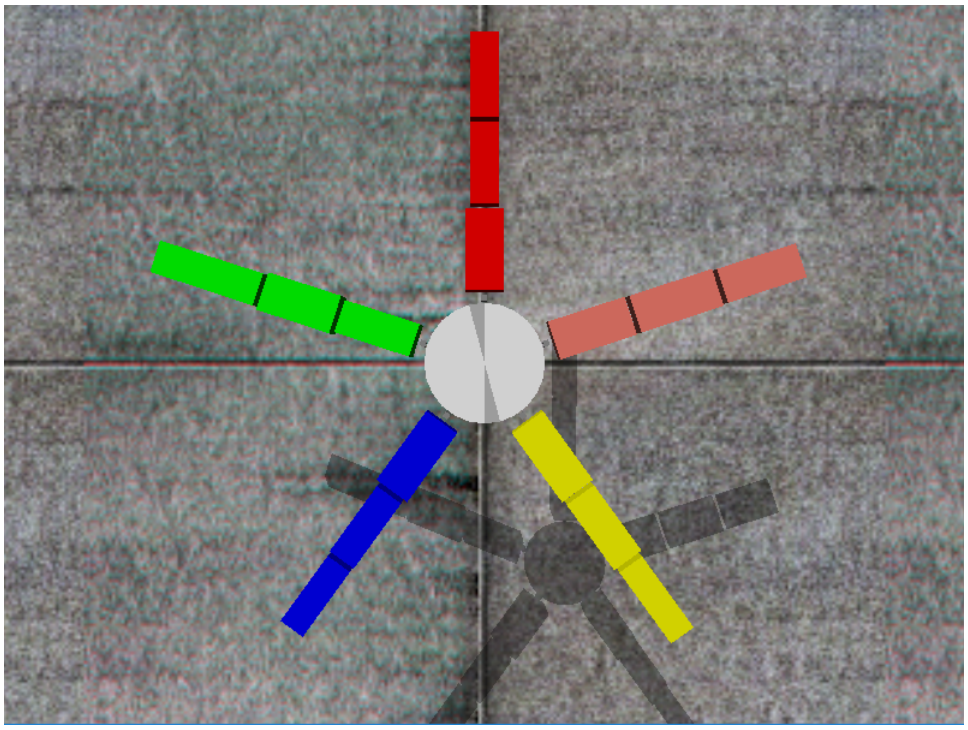

Figure 4.17: Best hand to grab and lift medium spheres.

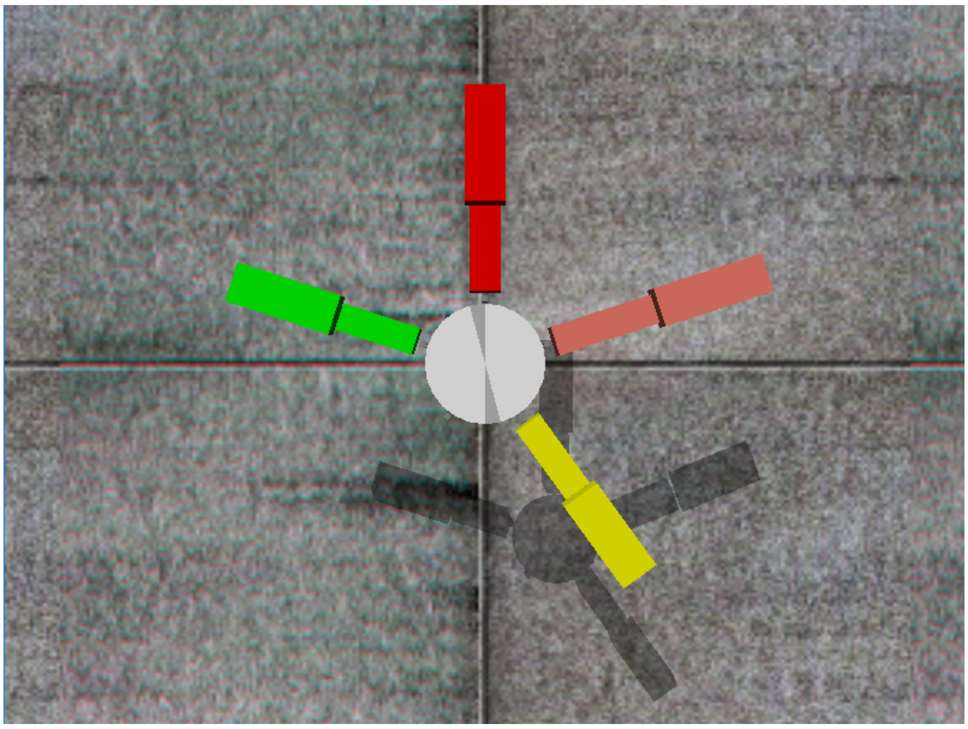

Figure 4.18: Best hand to grab and lift small spheres.

Chapter 5

Robustness Test of Evolved Morphologies

After the main evaluation experiment, each evolved hand was put through five different scenarios to test their robustness. These scenarios are useful to study how the previously evolved hands behave in different scenarios from the ones they were evaluated in. For that, we devised five different scenarios to test each hands' overall performance.

Each scenario has random noise applied to the object's initial position and rotation. This noise consists in small rotations of the object's x axis (0 to 1 degree) and of the object's position (0 to 1 mm). These small deviations in rotation and positioning help simulating how an object is placed in the real world.

Scenario 1 - Angles: *Each hand was evaluated with four samples, each representing one of the rotation angles along the X axis (0^o; 45^o; 90^o; 135^o). In this scenario to each of those angles a small deviation in its rotation is applied (0 to 1 degree) as well as to its position (0 to 1 mm). Each angle is tested one hundred times. Spheres were only subjected to random deviations on their positions.*

With noise applied to both the rotation and positioning of the object, we can simulate real life positioning of an object, as such we can prove if the evaluated hands can in fact successfully grab the object they were evaluated to grab and lift during the main evaluation experiments.

During this scenario, all hands that that had the highest fitness when grabing the large capsule object during the main evaluation process, continued to grab and lift the capsule even with the introduced random noise in rotation and positioning, with 100% success rate.

A pattern can be observed with medium evolved cylinder hands through this scenario in that most hands have more success grabbing the first angle, followed by the second angle, and the fourth angle coming in third, with the the third angle coming in fourth with 0% of success across all those hands (see fig. 5.1), which as we concluded during the main evaluation process results, is due to the angles being in different orientations.

The third angle is the most difficult angle to grab and lift the object in, due to its complete opposite orientation when compared to the other angles used. So, the two hands that could grab and lift the cylinder object in this angle, have the worst performance in the other angles (see fig. 5.2).

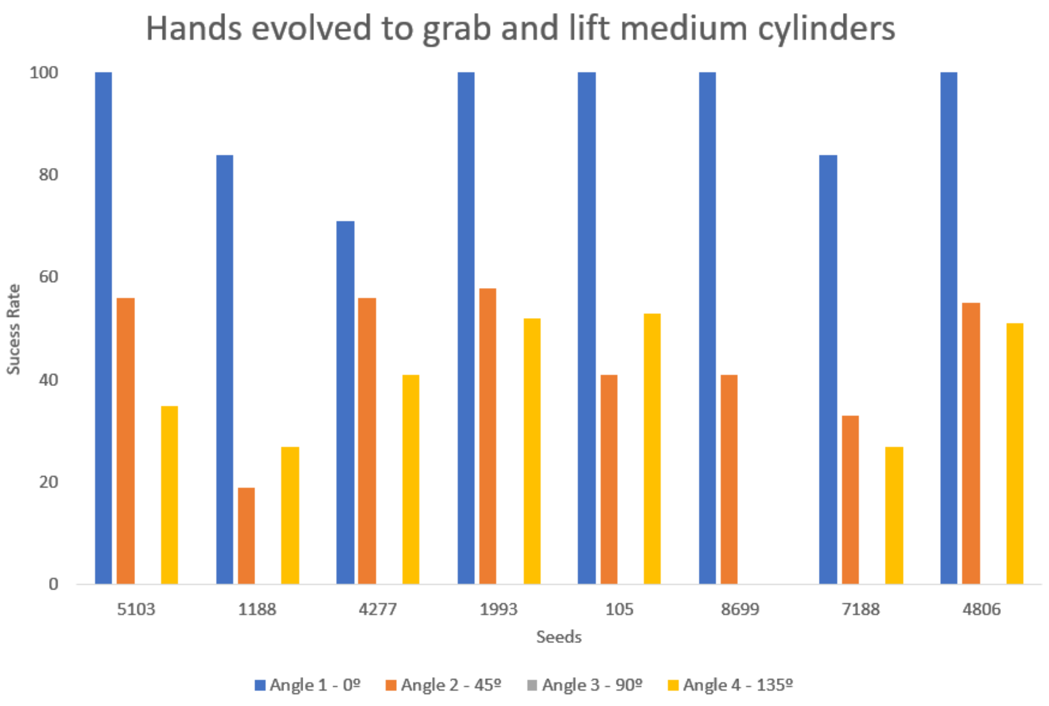

FIGURE 5.1: Visualizing the pattern emerging when the object is a medium cylinder.

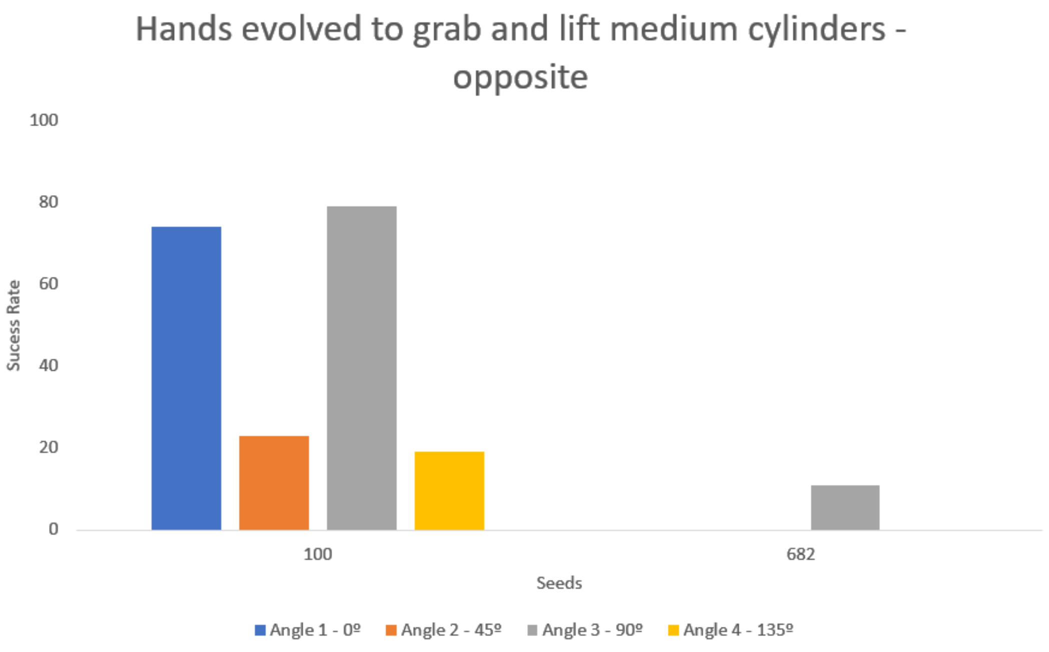

FIGURE 5.2: Visualizing the opposite pattern emerging when the object is a medium cylinder.

When we experimented with hands that were evaluated to grab and lift large rectangles, we found that two of the hands that didn't perform as well as the best one during the main evaluation process (hands that had a worse fitness value), adapted better to the random noise introduced in this scenario, the same happened with hands that were evaluated to grab and lift medium rectangles.

Contrary to what we thought, not all hands that were evaluated to grab and lift spheres during the main evaluation process could get a 100% success rate. This can be explained by the fact that the hands that could not grab and lift, were not using all the available positions in the palm of the hand, while those that could grab and lift spheres with 100% success rate used every available position in the palm of the hand.

With this scenario, we can conclude that the best hands are not always the ones that grab the object in specific angles and positions. By simulating possible real life noise, we can identify which hands have a better chance at completing the task outside a simulated environment.

Scenario 2 - Different Solids: *During the main evaluation process each hand was evaluated to grab and lift one specific object, however it is possible that hands can grab and lift different objects even if they were not explicitly evaluated to do so. This scenario tests a hands versatility by using a different object than the one it was evaluated to grab and lift during the main evaluation process. We limited this scenario to only grabbing and lifting objects of the same size. As an example, a hand evaluated to grab large capsules was only tested with large rectangles, large cylinders and large spheres.*

During this scenario, the best hands that were evaluated to grab and lift the large capsule object, adapted perfectly into grabbing and lifting the large sphere. This can be explained by the hands having a similar block placement, as these hands much like the hands that were evaluated to grab and lift spheres, use all available positions in the palm of the hand, and had three blocks in all positions. These best hands also had a 68.5% success rate when grabbing and lifting large cylinders.

Besides these best hands, there were other two hands that adapted perfectly in grabbing and lifting the cylinder. Both of these hands share a similar block placement as hands that were evaluated to grab and lift cylinders, with three pairs of blocks, one pair opposing the other two(see fig. 5.3).

It's interesting to note that, during this scenario, one of the best large capsule evaluated hands with fifteen blocks could adapt perfectly to grabbing and lifting the large rectangle. This is interesting because, all of the best large rectangle evaluated hands used twelve or fewer blocks, with those that used more blocks having a worse performance. This could mean that if those hands had continued to evolve to possibly use fifteen blocks, they could have had better performance.

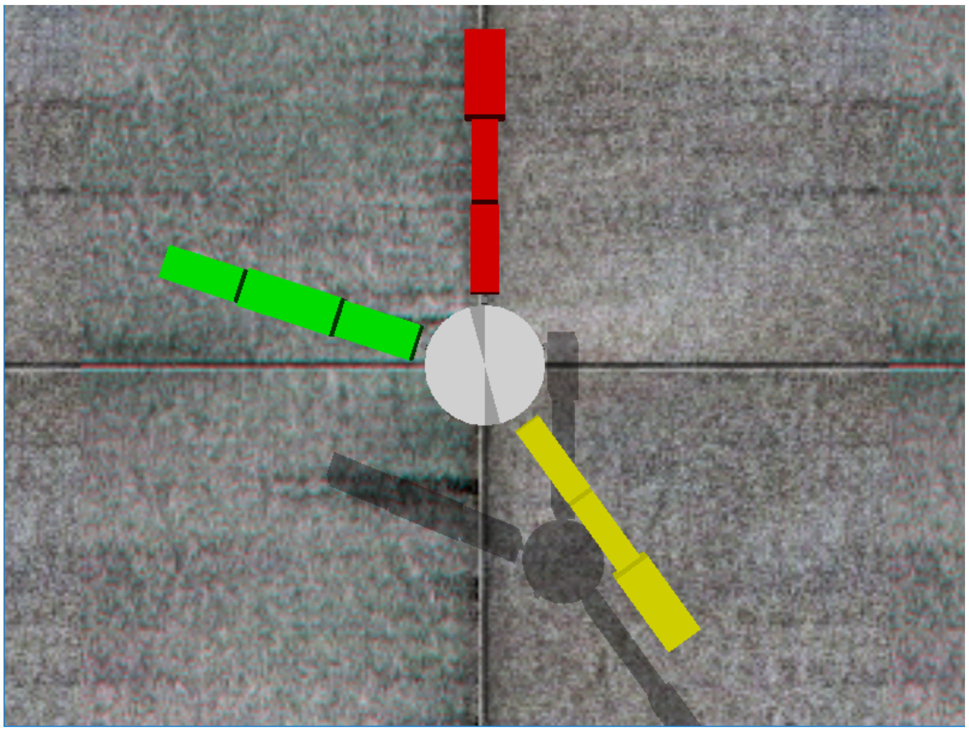

Figure 5.3: One of the hands evaluated to grab and lift the large capsule object that could adapt to large cylinders. As we can see it has a similar morphology related to hands evaluated to grab and lift cylinders.

There was no evolved hand that had a 100% success rate when grabbing and lifting large or medium cylinder objects during the main evaluation process, but during this scenario the best large and medium evolved hands had success rates of 99.5% when grabbing and lifting the large capsule, and 99% when grabbing and lifting the medium capsule. These hands also had a success rate of 82% when grabbing and lifting large rectangles. However no evaluated hand could, during this experiment, grab and lift any sphere size.

No hand evaluated during the main evaluation process to grab and lift the rectangle object could successfully adapt to grabbing and lifting capsules or cylinders, with a rate of success across all sizes of 23% for capsules and 18% for cylinders.

However, most of the best hands evaluated to grab rectangles, adapted with a success rate of 100% to grab and lift large and medium spheres. These rectangle hands made use of all available positions in the palm of the hand (see fig. 5.4), much like hands that evolved to grab and lift spheres. This explains how they reached 100% success rate.

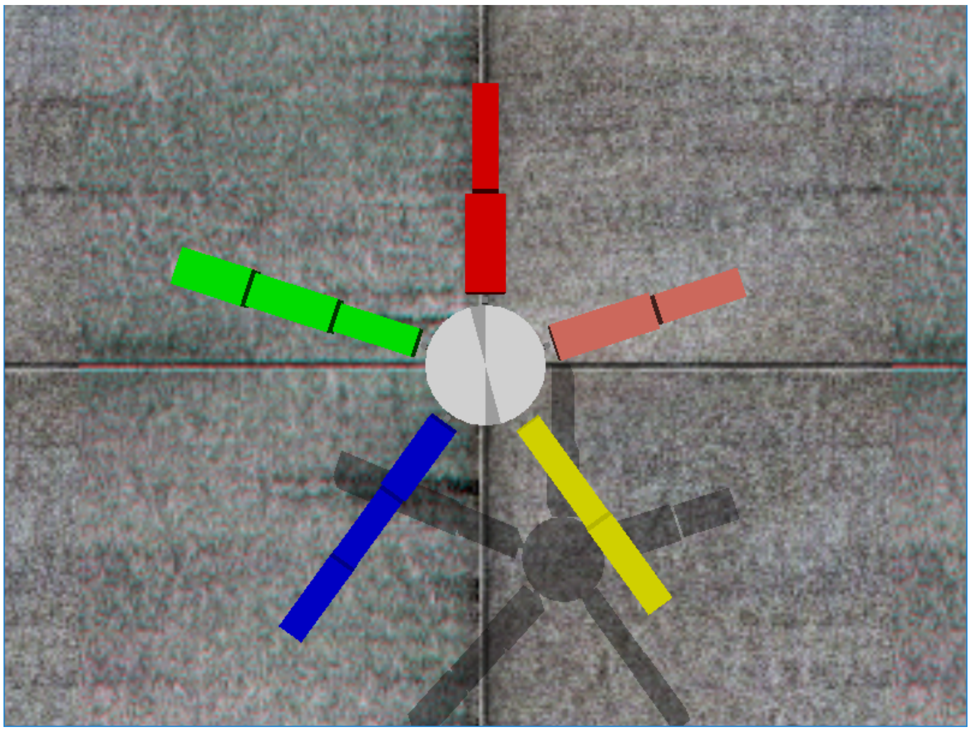

FIGURE 5.4: One of the hands evaluated to grab and lift large rectangle objects that adapted into grabbing and lifting large spheres.

Hands evaluated during the main evaluation process to grab and lift spheres, showed to be incapable of adapting to any of the different solids that were used.

Because hands evolved to grab spheres tend to use all positions available in the palm of the hand, this leads to at least one block in each position. However, most hands evaluated to grab and lift spheres, always use more than five blocks, this leads to a possibility of secondary or tertiary blocks that can interfere with how an object is grabbed.

With this scenario we can conclude that, even hands with bad performance in grabbing and lifting their object, could potentially be used to lift and grab other objects in the same size category if they share a similar morphology with the best hands of the other object.

No hand could adapt to another object when the object size was small. With the exception of one hand that had a success rate above 50%, with 53%.

Scenario 3 - Random Angles: *Unlike the first scenario where only a small deviation over the four main samples was applied, in this scenario a completely random angle is generated from 0 to 360 degrees.*

The goal of this scenario was to test if the hand could grab and lift the object over angles it was never evaluated with during the main evaluation process.

In this scenario, hands that had a 100% success rate in grabbing and lifting the large capsule during the main evaluation process, also had a 100% success rate in grabbing and lifting at every random angle generated, and as expected, the difficulty in grabbing and lifting the medium capsule object led to a bad performance in this scenario.

The best hand that was evaluated during the main evaluation process to grab and lift small capsules, in this scenario placed in fifth with 64% success rate, as such we can conclude that, this hand evolved specifically to grab and lift in only the four angles used during the main evaluation process.

There are other hands that did not performed as well in the main evaluation process, that adapted in grabbing and lifting the small capsule in random angles, with a 79% success rate.

During the main evaluation process, the cylinder objects were the most complicated to grab and lift, so it's understandable why hands evaluated to grab and lift cylinders had such a poor performance in this scenario, with only two hands managing a success rate above 50%, at 53% and 55%.

None of the best hands from the main evaluation process with a rectangle as an object had the best performance in this scenario. The best large rectangle evaluated hand had a 65% success rate, however there were hands that had a higher success rate in this scenario, with the highest success rate being 88%.

While most hands evolved to grab and lift spheres had 100% success rate, in this scenario, much like what happened with hands evolved to grab other solids, not all hands with 100% success rate in the main evolution had a 100% success rate in this scenario.

While rotating the sphere has no impact due to its symmetric nature, the random positioning is enough to make it so the hand cannot grab the sphere as it did during the main evaluation process.

With this scenario we can conclude that, while hands evaluated to grab and lift an object, could do it well in most samples, it doesn't necessarily mean that the evolved hand is capable of grabbing and lifting the same object in a completely different angle.

Scenario 4 - Size increments: *In this scenario, we generated 100 different sizes from the smallest size of an object to its biggest size, divided them in groups of tens and tested how many sizes a hand could grab and lift in each group. Each increment was calculated by subtracting the smallest size from the biggest and then dividing by one hundred.*

With this scenario, we can detect at what size hands start to be effective in grabbing and lifting objects.

Hands were evaluated to grab and lift objects in three different sizes. However there is a noticeable pattern in some situations where the best hands evaluated to grab and lift large capsules, are effective in grabbing and lifting capsules with 60% of the original large capsule size (see fig. 5.5). This pattern can be observed in the rest of the large objects to various degrees of success, even in objects where a hand could not grab and lift the object with 10% success rate in the main evaluation process, such as the rectangle (see fig. 5.6).

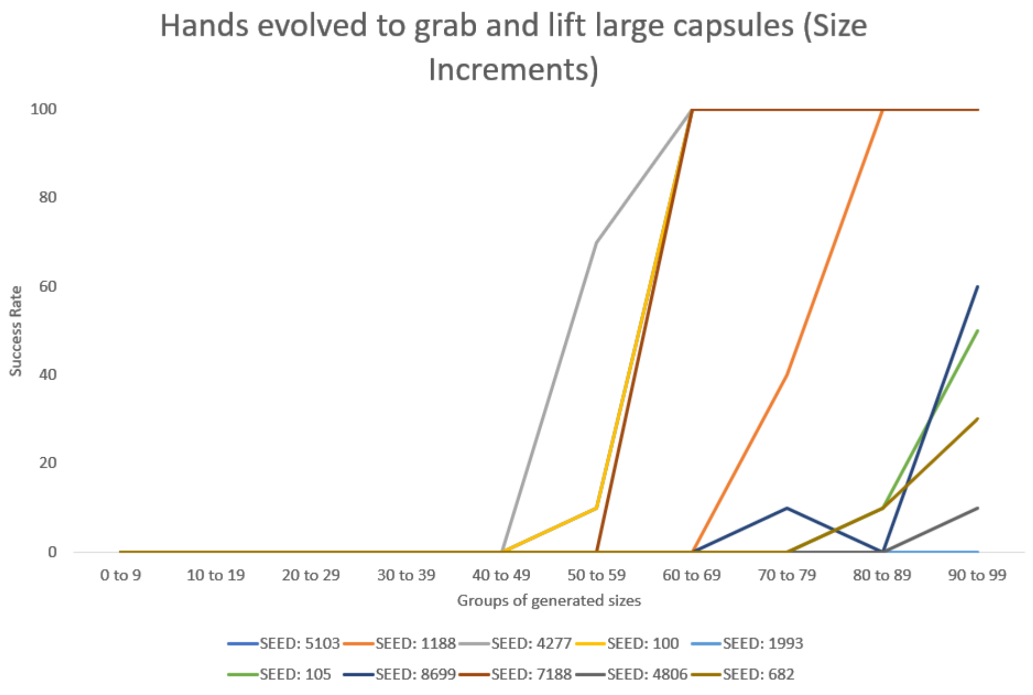

FIGURE 5.5: The pattern emerging in hands evaluated to grab and lift large capsules when the size of the capsule is incremented.

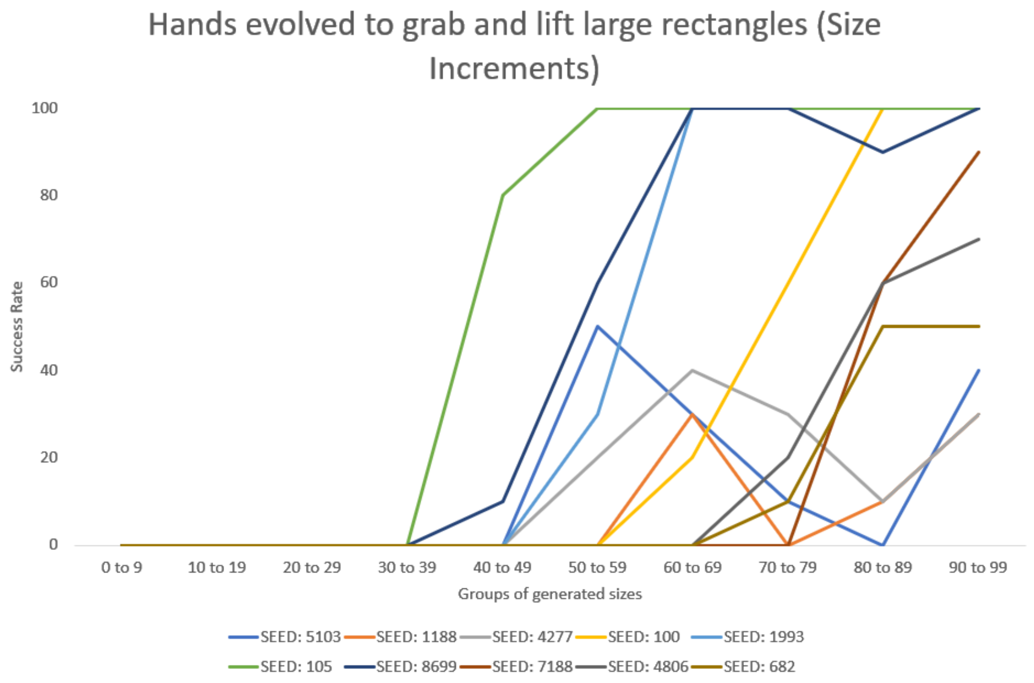

FIGURE 5.6: Much like hands evaluated to grab and lift large capsules, these hands are able to grab rectangle sizes that are half the size of a large rectangle.

Experiments with medium objects also had a pattern of when hands could grab and lift the object (see fig. 5.7 and fig. 5.8). In experiments with medium objects, hands were found to be effective when the object size is about 10% bigger than the smallest object, and continue to be effective until the object reaches about 59% of the size of the smallest object, before starting to decline.

There are two exceptions however, hands evaluated to grab and lift medium sized capsules could not grab and lift any size increment during this scenario. While hands evaluated to grab and lift medium sized spheres could grab and lift most size increments in 50% of the seeds.

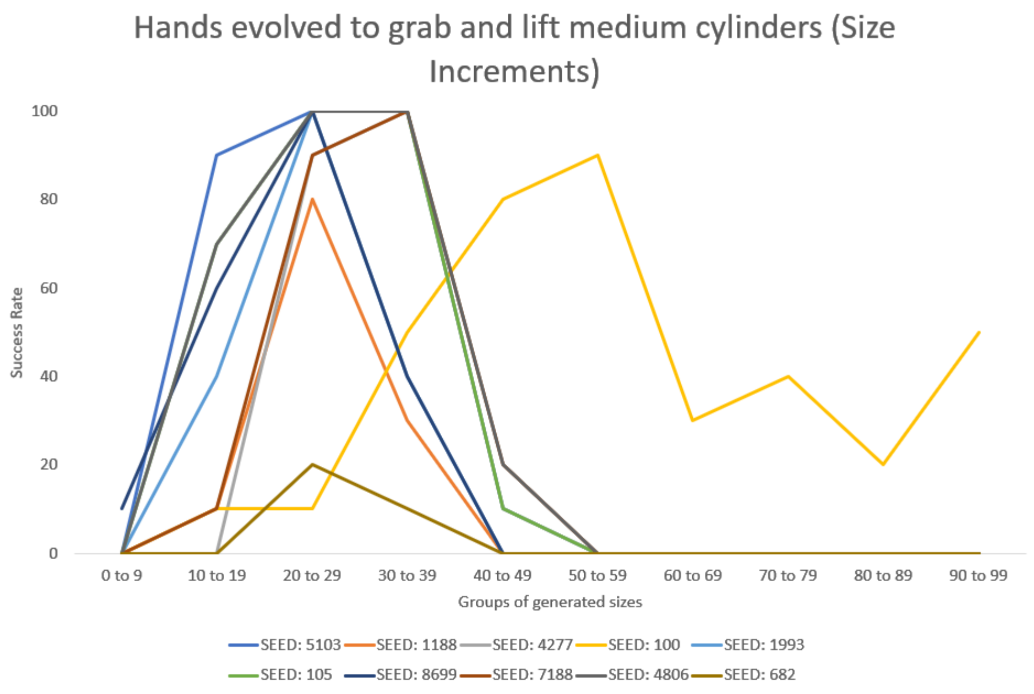

FIGURE 5.7: Hands evaluated to grab and lift medium cylinders are able to lift cylinders 10% bigger than the smallest size up to cylinders 49% bigger than the smallest size.

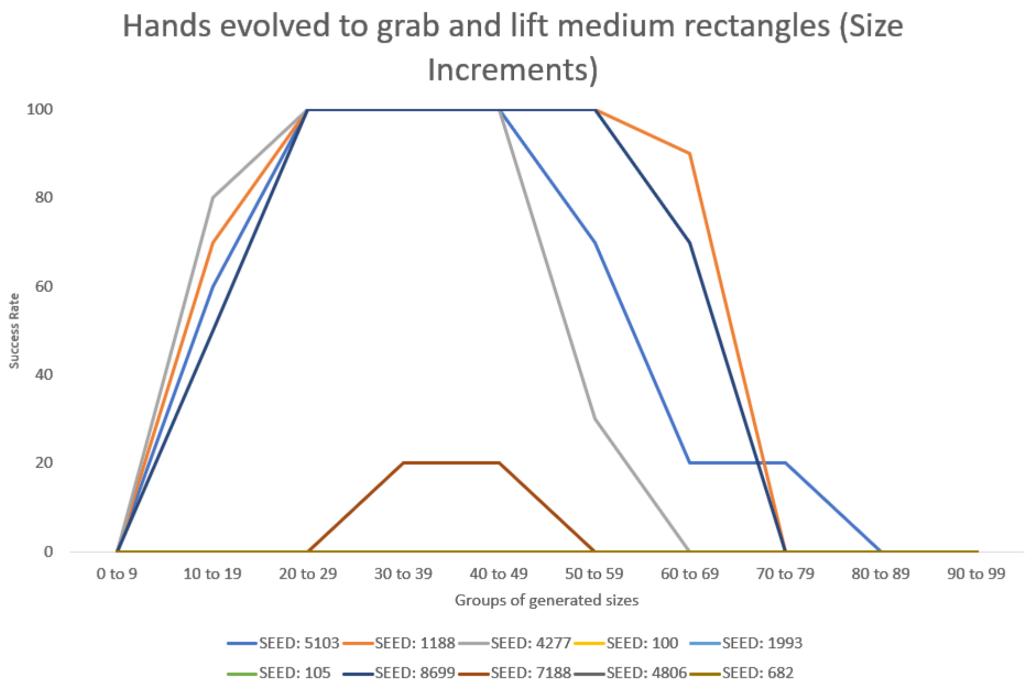

FIGURE 5.8: Hands evaluated to grab and lift medium rectangles are able to lift rectangles 10% bigger than the smallest size up to rectangles 50% bigger than the smallest size.

Scenario 5 - Different Sizes: *In this scenario, each hand tries to grab and lift the same object it was evaluated to grab and lift in the main evaluation process, but in a different size. So, for example, a hand that was evaluated to grab and lift a large capsule, tries to grab and lift a medium and a small capsule.*

In this scenario, 73% of the 120 evaluated hands could never grab and lift a different size of the same object. Six hands that were evaluated to grab and lift medium capsules could grab and lift small capsules, but their success rate was low (from 6% to a maximum of 36%).

There were two hands that were evaluated to grab large cylinders with a success rate of 96% and 97% when grabbing medium cylinders, and they both share the same block positions and the same block number and placement.

Spheres could more easily adapt to different sizes with several 100% success rates, most of them when a hand evaluated to grab and lift medium spheres had to grab and lift the large and the small sphere.

These scenario results are in accordance with those of the previous scenario (Size Increments). In the previous scenario, we observed that hands evolved to grab and lift medium cylinders and medium rectangles could not adapt into grabbing and lifting small or large objects. The same behavior is present in hands evolved to grab and lift large objects. In this scenario, hands evolved to grab and lift medium objects could not grab and lift small and large sized objects, the same is valid to hands evolved to grab and lift large objects thus confirming the previous scenario results. Furthermore, hands that were evolved to grab and lift medium sized spheres that could adapt to grabbing and lifting all sizes, were the same hands that could in this scenario grab and lift the different sizes.

As previously mentioned in the results, the best hands tended to use similar morphologies to grab and lift objects during the main evaluation process, this of course leads to specific solutions. In conjunction with a tendency to use less blocks the smaller the object is. This explains why most hands performed poorly in this scenario, as hands evolved to grab and lift bigger objects would have too many blocks to grab and lift smaller objects.

Chapter 6

Conclusions

In this dissertation, we demonstrated that it is possible to apply genetic algorithms to create specific morphologies that specialize in completing a specific task. We were able to evolve a gripper that could grab an object from the ground, lift it and hold it for a defined amount of time. We evolved a gripper for each object type and size, giving us a total of 120 grippers by the end of the evolutionary process. This evolutionary process consisted in evolving 100 grippers along 250 generations for each object and object size combination. Each gripper was evaluated in a simulation where fitness points were attributed according to the gripper's performance when executing the task. The simulation consisted in simulating an environment where the generated gripper would try to grab and lift an object, and hold it until the simulation finished. All objects with the exception of the sphere, were evaluated in four samples with each sample starting the object at a different angle, thus evolving a more robust gripper capable of grabbing and holding the object in more than one angle. The sphere was not evaluated in different samples because, due to its symmetry, a different angle would not have a different outcome.

At the start of each generation, the best ten grippers of the previous generation were chosen to be copied, with each copy having a probability of having its blocks mutated in their height and length parameters, adding or removing blocks, or copying a complete block structure to another available position at the palm of the hand. This process was repeated until the last generation.

Hands evaluated to grab the sphere object had a 93% rate of success across all seeds. While the hands evaluated to grab and lift capsules and rectangles had much lower rates of success across all seeds, 17% and 3% respectively. Hands evaluated to grab and lift the cylinder object were the only ones that could not successfully grab and lift the object across all seeds.

These results can be explained by the object's complexity, the sphere is a much more symmetric object, with its radius barely crossing the hands radius, while the other objects all had a bigger part outside the hands radius. Depending on how the hand was grabbing the object, and because of the object's symmetry, its weight would be distributed accordingly thus creating an imbalance, making it harder for the hand to lift the object without dropping.

Another explanation for the low success rate in all objects except the sphere was the third sample used during evaluations. This third sample is a rotation of 90^o over the objects X axis, which is in a completely different orientation from the first sample at 0^o. Therefore most hands evolved to grab and lift angles much closer to that first one, due to the similarities with the second and fourth angle, indirectly making the third angle much more difficult to grab leading to a worse success rate. The sphere was evaluated in only one sample due to its symmetry.

After we completed the evaluation, we tested the robustness of all hands in five new scenarios to assess how well these hands would perform in scenarios not present in the evolutionary process.

The first scenario we experimented with the same angles used during the main evaluations experiment with added noise to simulate real world conditions, such as positioning and rotation. In this experiment, we found that the best hands to grab and lift large capsules could do it even with noise, while some of the best hands to grab large spheres had problems grabbing the sphere with this extra noise, although the hands that had such problems were the ones that didn't use all the available positions where blocks could be attached on the palm of the hand therefore creating a hole in one of the sides from which the sphere could fall from.

In the second scenario, we tested how well hands would grab different solids from the ones they were evolved with. Hands that originally evolved to grab and lift large capsules could adapt to grabbing and lifting large spheres. Some of these hands that shared a similar morphology with hands that evolved to grab and lift large cylinders had a 100% success rate in adapting to grabbing and lifting large cylinders. Hands that evolved to grab and lift large and medium cylinders adapted to large and medium capsules with more than 90% success rate. Hands that evolved to grab and lift large cylinders also had a 82% success rate when grabbing large rectangles, however they could not adapt to grab spheres of any size. Rectangle evaluated hands could only adapt to spheres, while sphere evolved hands could not adapt to grabbing any object.

In our third post evaluation scenario, we experimented with completely random angles to check whether hands could grab and lift objects outside the angles they were evolved with. The best hands evolved to grab and lift large capsules and most hands evolved to grab and lift spheres (70%), completed this experiment with a 100% success rate. There were some cases where the best hands did not have the best performance which means that, while those hands were the best in grabbing the objects in specific angles, they could not however grab the same object in a random angle that was different from the ones originally used during main evaluations.

In the fourth scenario we experimented with size increments, starting with smallest size used during normal evaluations, incrementing its size until it reached the largest size used during the main evaluations. There were some noticeable patterns with hands evaluated to grab and lift large objects normally starting to grab the later fourth of sizes, but not sizes prior to those. While hands evaluated to grab and lift medium objects would start to grab and lift objects when they were 10% bigger than the smallest object size, reaching a peak of success in grabbing and lifting objects 50% bigger than the smallest object, and then declining.

The fifth and final scenario consisted in experimenting different hands of the same object with different sizes of that object. Most hands across all seeds (73%), had zero success rate in grabbing different sizes of the same object they were evolved to grab and lift. Although there were two hands originally evaluated to grab and lift large cylinders that adapted to different sizes with a success rate bigger than 95%. Spheres adapted easily to different sizes, but even then not all hands were able to adapt to grabbing and lifting different sizes of the same object.

After running these experiments, we can conclude that the limitations imposed to create and modify blocks, lead to hands that have a hard time grabbing and lifting smaller objects.

To evolve the morphology of a gripper for a task there needs to be some boundaries in place so the sizes of the blocks are not too extreme (too small or too big, too wide or too thin). The simulation must also take into account the most important variables that can affect experiments outside of a simulated environment, so that the evolved hand resembles a hand that can be produced and used outside simulations. Taking into account the most important variables present in the real world, and applying them in the simulation, is a method of reducing the size of the reality gap proposed by Jakobi (1998)(1997).

After these experiments, we can conclude that genetic algorithms are suited to evolve gripper block morphology. Because this type of algorithm is based on biological processes, our solutions were very similar to hand morphology found in nature, such as our own hand with five positions each having three blocks, or hands evaluated to grab and lift cylinders that resemble a bird's foot with two pairs of blocks opposing one pair. However only 28% of the evolved solutions were capable of grabbing and lifting the object without letting it fall. This poor success rate can be attributed to the combination of very different samples and not due to the genetic algorithm failing to produce a functioning hand.

There is a tendency for hands to share block piece numbers and positions. We can observe this with hands that evolved to grab and lift spheres, and hands that were evaluated to grab and lift large capsules. The best hands for these two objects all share the same block numbers and positioning, with hands that don't share this morphology performing worse. As such we can conclude that there is homogeny present in the morphology of our evolved hands.

Our initial idea was to evolve hands using our simulator and to later print them using a 3D printer so we could test them in a real life scenario. This idea was scrapped when we couldn't adapt the ODE framework to work alongside Conilon (Silva et al., 2011). Conillon is a computing platform developed by BioMachines-Lab that allows tasks to be computed on computers that share the same network. Conilon would have helped accelerate the evolutionary process by distributing simulations throughout other computers connected to the same network, thus simulating multiple hands at the same time. We managed to distribute simulations locally by assigning them to threads. With this solution, each seed took an average of five days to complete.

There was also some initial difficulty using the ODE framework, due to how vast the framework is. Units are free for the user to interpret them which created some problems as we initially thought that units were already in SI. We chose to measure sizes in centimeters, and as such we had to scale our objects to accommodate for this unit to work. We also had to change the torque applied in objects, their weights, the world step (how many seconds per tick the world moves), and other ODE parameters so the simulation could run.

This was a major problem we faced with ODE. Changing the value of a parameter indirectly implied changes in other parameters due to the way ODE is built. This made it harder to find the correct parameters to run the simulation.

Adapting the fitness function took longer because of the time the simulation took to complete. To be sure the fitness function was working or not, we needed at least one evaluation of each object, which meant twelve evaluations, with some evaluations taking up to a full day to complete. For that reason, we could not use a better fitness function in our experiments that could prevent false positive results.

ODE has four primitive objects (capsule, rectangle, cylinder, sphere), and there is also the possibility to import complex ones. We implemented this in our simulator, but found that complex objects required some work before we could use them in our simulations. Complex objects could be imported through .obj files, however because there is no standard for this type of file, we had to make sure the objects we wanted to import followed ODE's specification for this file type. This meant we had to clean up files that had unnecessary information.

After importing complex objects to ODE, and running some simulations, we noticed that the time it took to run these simulations was much longer than a primitive object. This of course is due to the number of triangles a complex object has. So with our hardware limitations, and complex object's evaluations taking much longer to complete than we had anticipated, we decided to discard them from our final evaluations.

We integrated in our simulator the creation of block pieces as we had envisioned them to be printed. Using the framework Processing, we were able to combine complex 3D pieces such as the fin, the hole in the middle to place the servo, and the joint parts that would connect to other joint parts either from an hand or from another block piece, into one final block piece.

To improve the evolutionary process, it would be beneficial to add a controller to each block. As it is, the hands follow a script to open and close, with this improvement, each block would open or close independently of others, making it so that for example, when a block was touching the object, it would stop pushing the object further, while other blocks kept closing until they too could touch it, thus grabbing the object in a more refined way. With controllers, the force applied in each joint could also adapt during the different phases of the script, more force could be applied during the "move up" phase of the script to make sure the object doesn't fall for example.

Adding to the suggestion above, coevolution could help the controllers to better adapt to the morphology of each block, while each block would also evolve based on the controllers.

Due to time constraints we could not perfect the fitness function so that the evolved hands could all grab and lift the proposed objects. Improvements over the fitness function could be, to differentiate when an object is grabbed and lifted but falls, from an object that never left the ground, with that we could discard solutions that could never lift the object from the ground.

One of the reasons for the poor success rates was the third sample. Due to its different orientation from the other samples, a large number of the evaluated hands could not lift or grab the object in this sample. One of the improvements would be to increase the number of generations to evolve a hand, so that the hands could adapt to grabbing and lifting all samples.

Similar to the suggestion above, we could separate each sample in different evaluations. This would lead to more evaluations, however after all the four evaluations, we could try to combine the four evolved hands to create a hand that could grab and lift the four samples.

Appendices

Appendix A

Hands evaluated to grab capsules: Fitness Graph

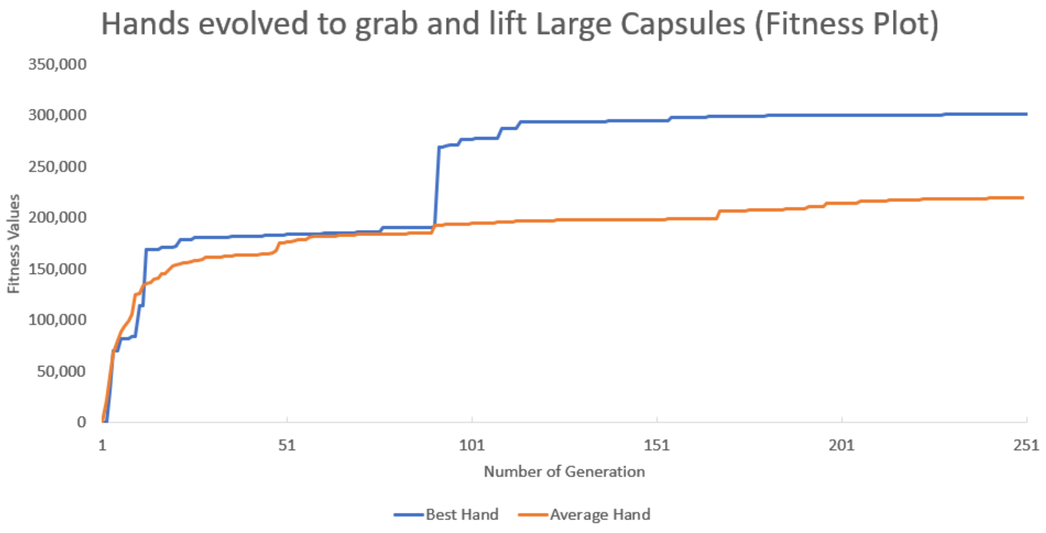

FIGURE A.1: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift large capsules.

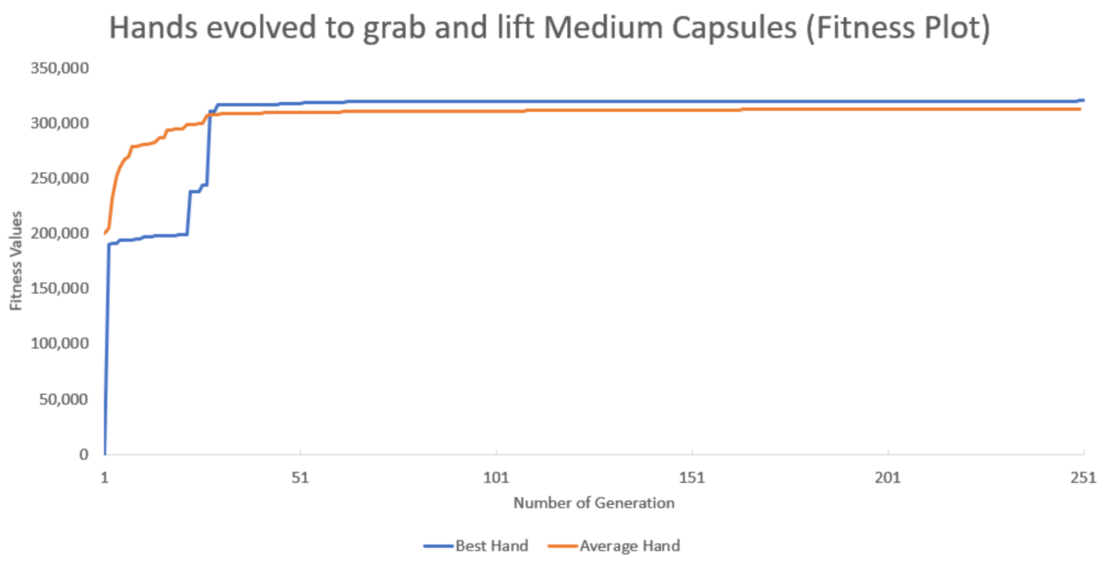

FIGURE A.2: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift medium capsules.

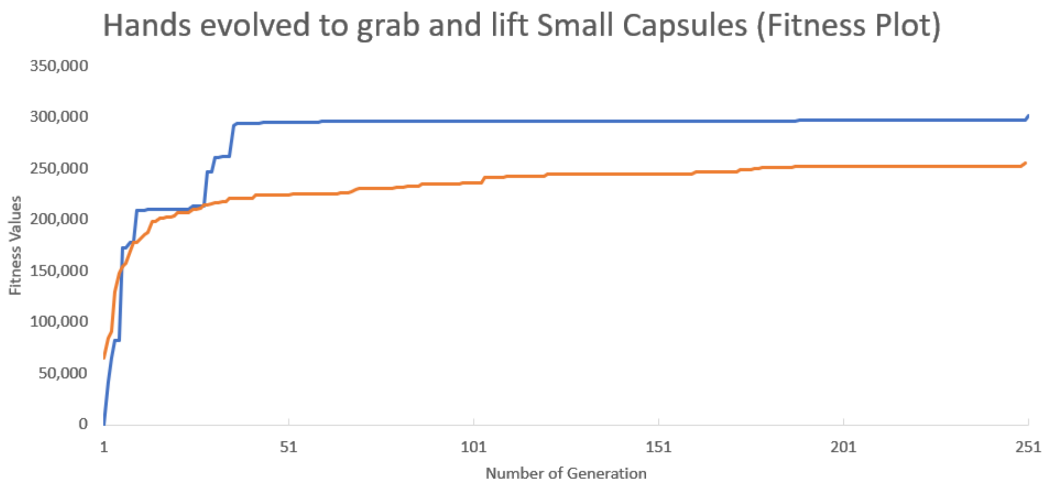

FIGURE A.3: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift small capsules.

Appendix B

Hands evaluated to grab cylinders: Fitness Graph

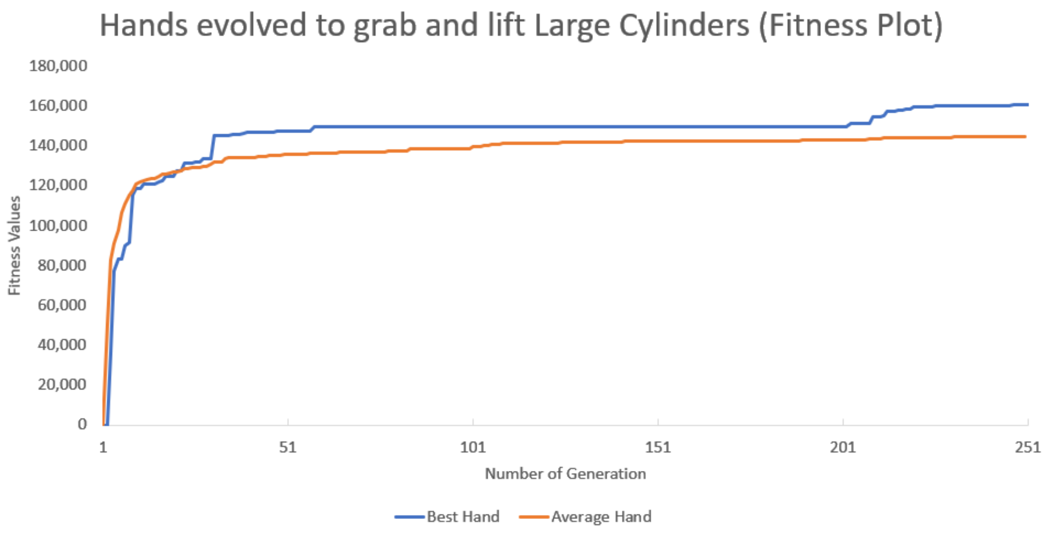

FIGURE B.1: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift large cylinders.

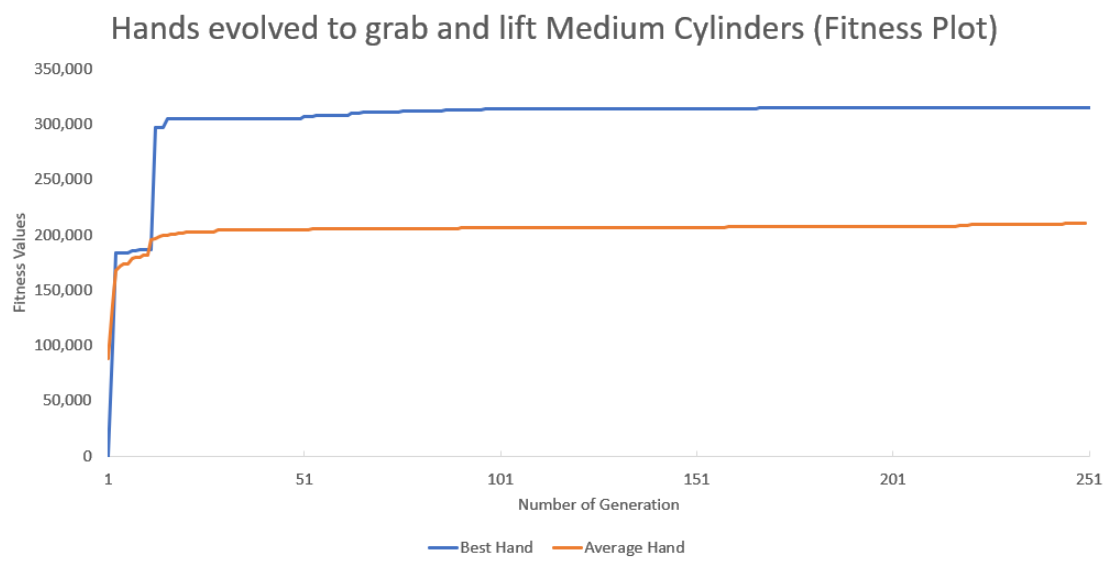

Figure B.2: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift medium cylinders.

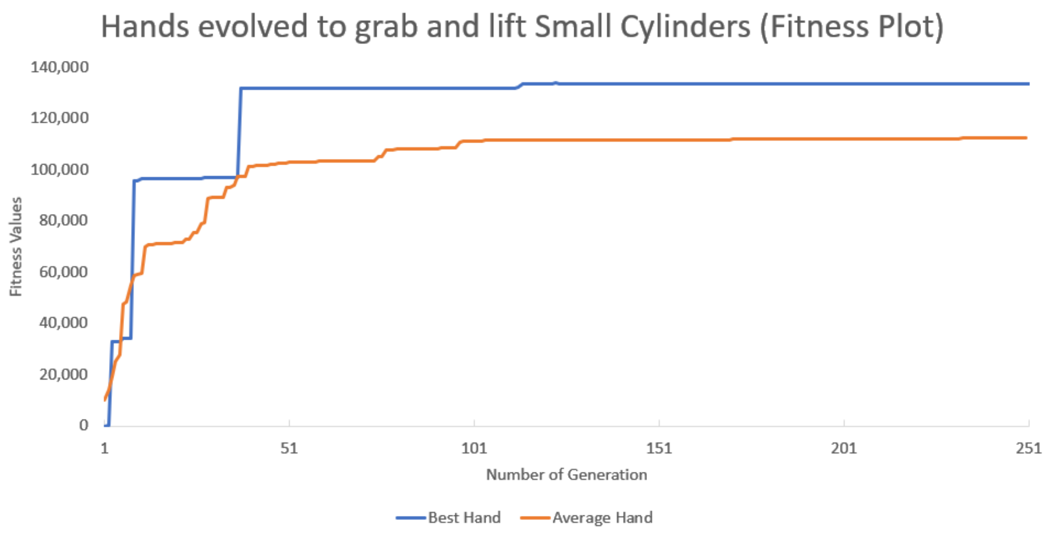

Figure B.3: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift small cylinders.

Appendix C

Hands evaluated to grab rectangles: Fitness Graph

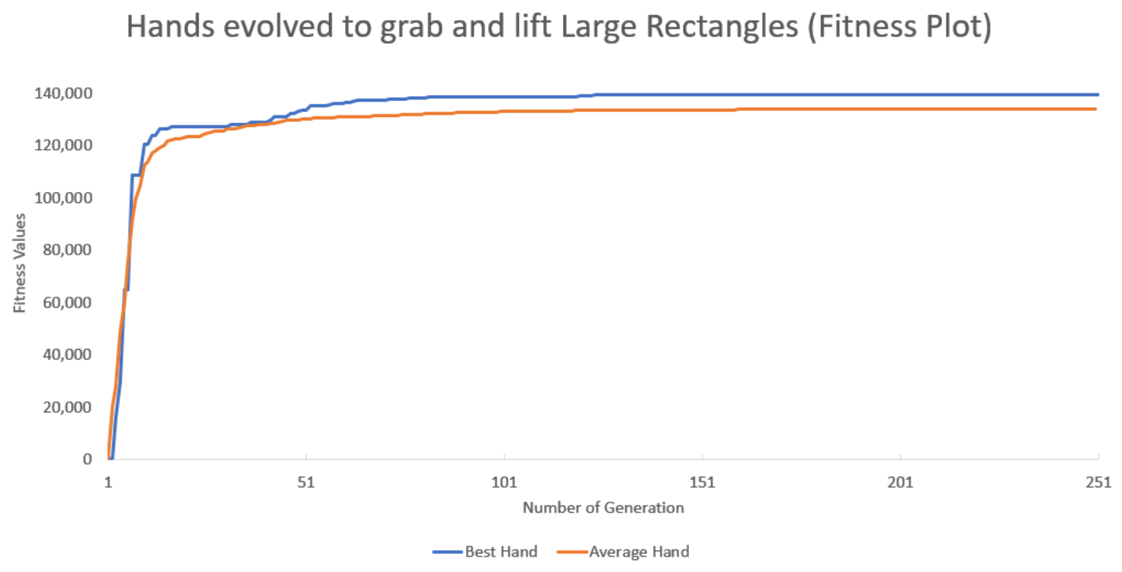

FIGURE C.1: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift large rectangles.

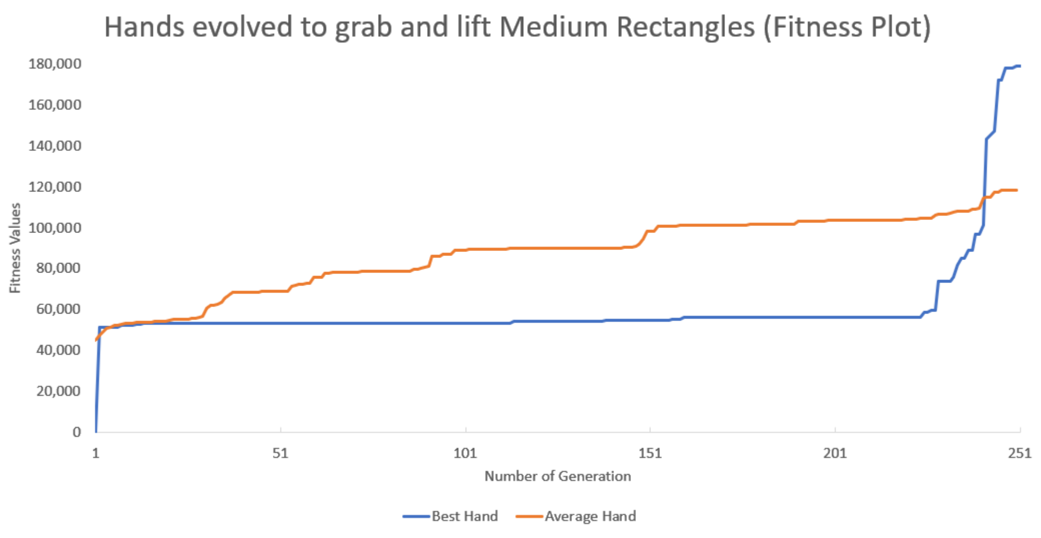

FIGURE C.2: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift medium rectangles.

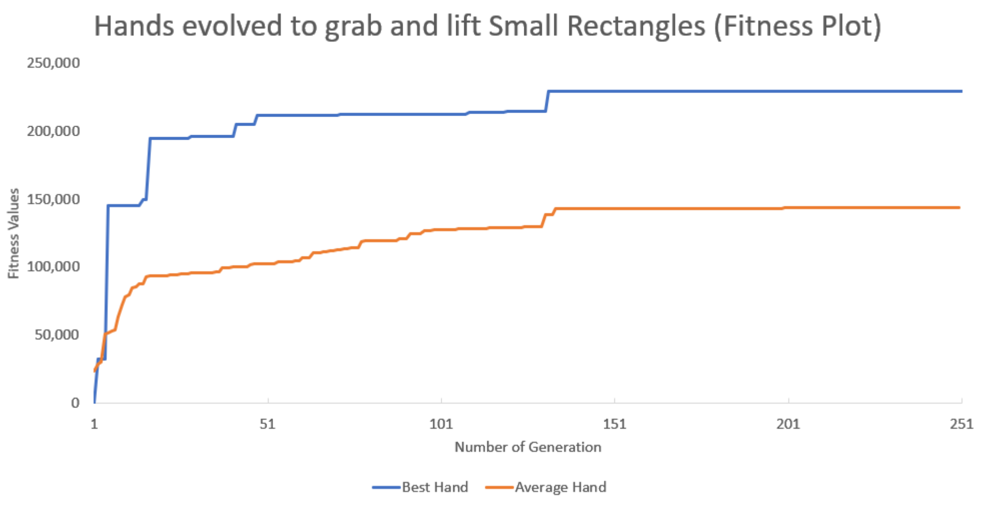

FIGURE C.3: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift small rectangles.

Appendix D

Hands evaluated to grab spheres: Fitness Graph

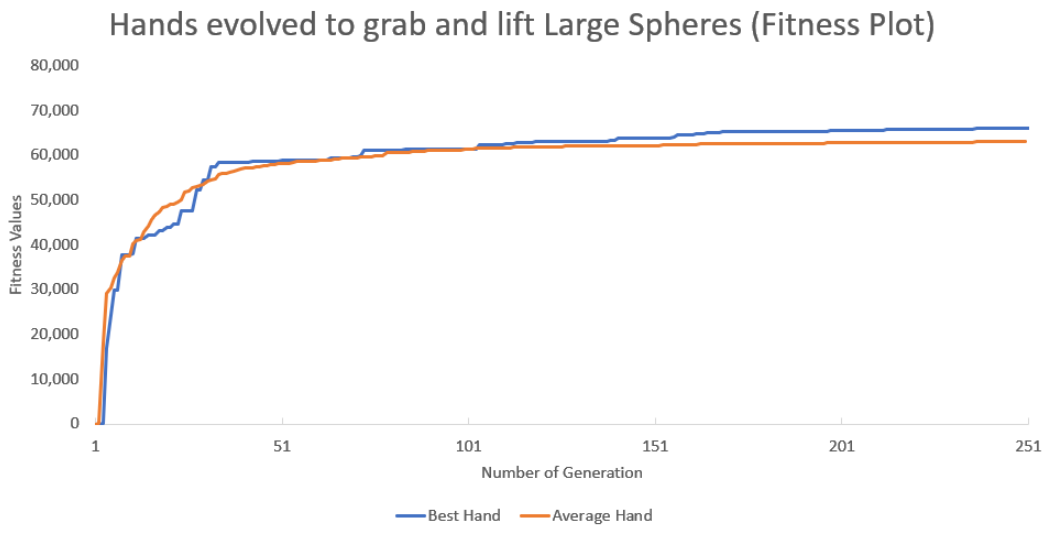

FIGURE D.1: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift large spheres.

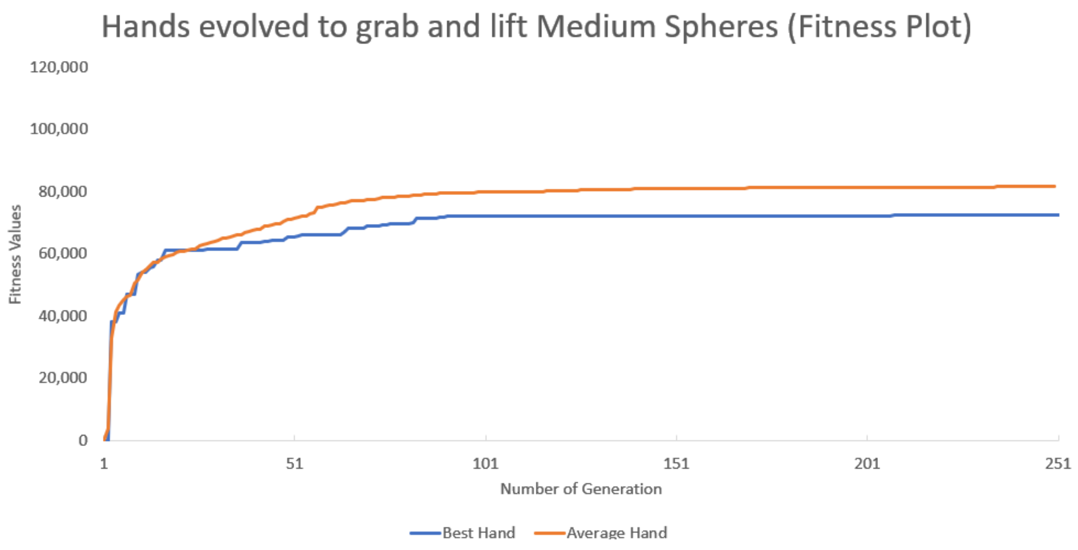

FIGURE D.2: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift medium spheres.

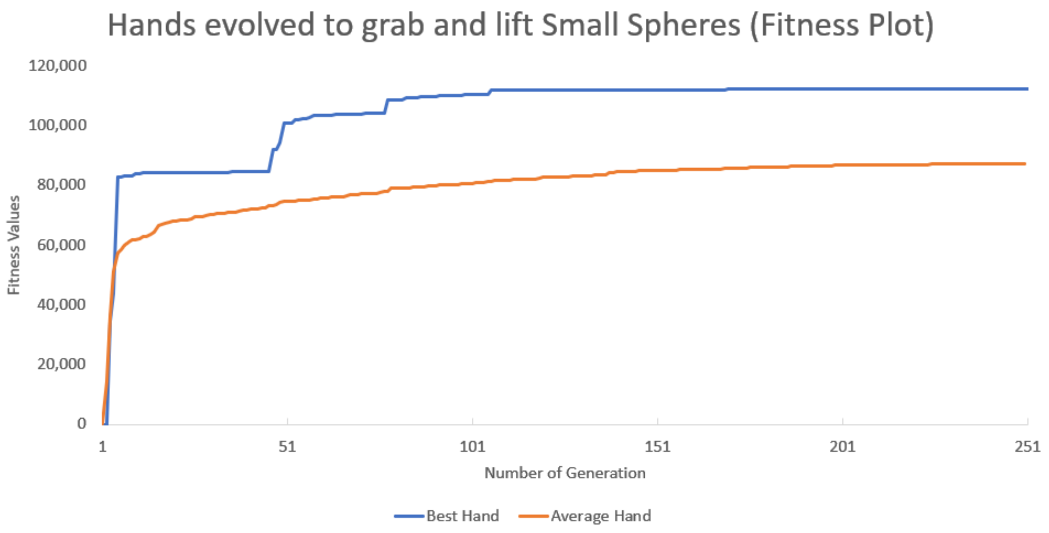

FIGURE D.3: Fitness graph with the average fitness values of each of the highest scoring morphologies of all 10 seeds, and the fitness values from the best morphology at each generation to grab and lift small spheres.

Appendix E

Sequence of a successful hand

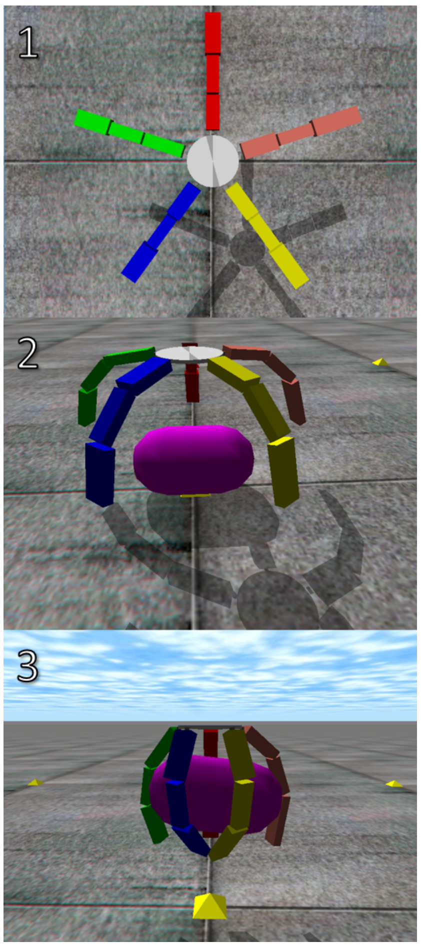

FIGURE E.1: 1) Opening Hand: Blocks rotate outwards over the hinge joints that connect them to the hand and other blocks; 2) After reaching their maximum rotation angle outwards over the hinges, blocks start to rotate inwards, thus closing the hand; 3) The hand successfully grabs the object and goes up, holding it until the simulation ends;

Appendix F

Example of an unsuccessful hand

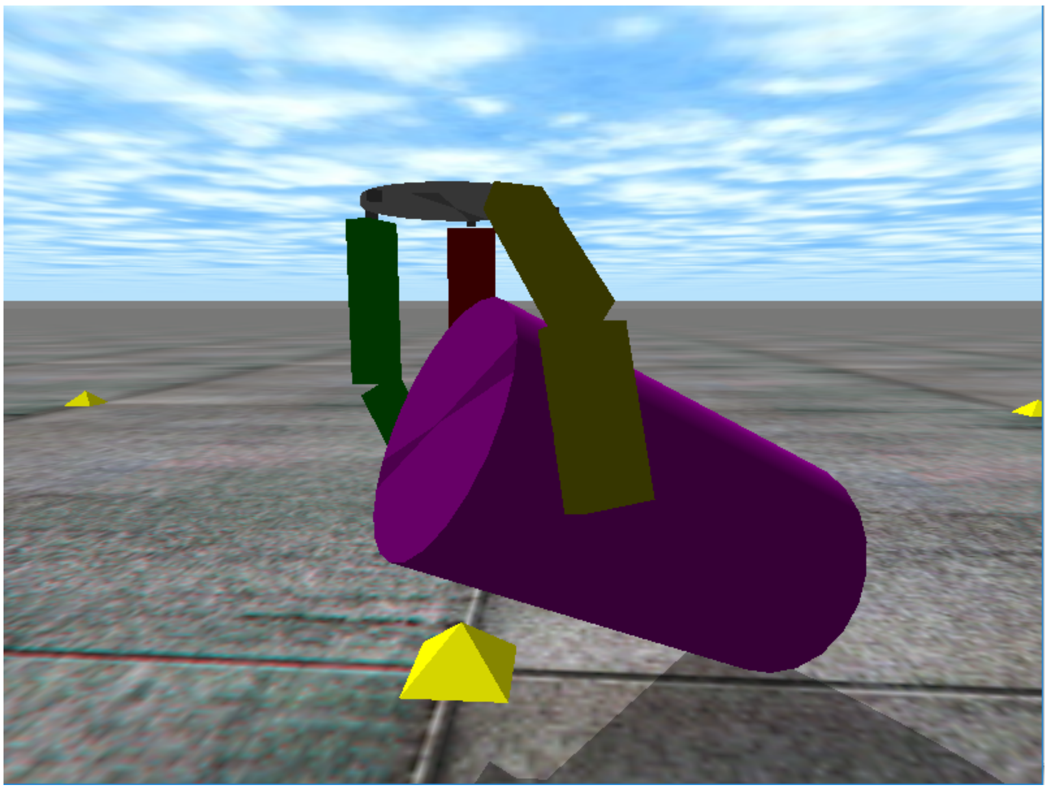

FIGURE F.1: Example of a hand failing to grab and hold the object until the end of the simulation;

Bibliography

Alvaro Bertelsen, Javier Melo, Emilio Sánchez, and Diego Borro. A review of surgical robots for spinal interventions. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 9(4):407–422, 2013.

S.I. Cho, S.J. Chang, Y.Y. Kim, and K.J. An. Ae—automation and emerging technologies: Development of a three-degrees-of-freedom robot for harvesting lettuce using machine vision and fuzzy logic control. *Biosystems Engineering*, 82(2):143 – 149, 2002. ISSN 1537-5110. doi: https://doi.org/10.1006/bioe.2002.0061. URL `http://www.sciencedirect.com/science/article/pii/S1537511002900619`.

George C Devol. Programmed article transfer, June 13 1961. US Patent 2,988,237.

D. Floreano and J. Urzelai. Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Netw.*, 13(4-5):431–443, May 2000. ISSN 0893-6080. doi: 10.1016/S0893-6080(00)00032-0. URL `http://dx.doi.org/10.1016/S0893-6080(00)00032-0`.

Dario Floreano and Laurent Keller. Evolution of adaptive behaviour in robots by means of darwinian selection. *PLOS Biology*, 8(1):1–8, 01 2010. doi: 10.1371/journal.pbio.1000292. URL `https://doi.org/10.1371/journal.pbio.1000292`.

Ben Fry and Casey Reas. Processing, 2001. URL `https://processing.org/overview/`.

Stephen Jay. Gould. *The Structure of Evolutionary Theory*. BELNAP PRESS, 2002.

John H. Holland. Outline for a logical theory of adaptive systems. *J. ACM*, 9 (3):297–314, July 1962. ISSN 0004-5411. doi: 10.1145/321127.321128. URL `http://doi.acm.org/10.1145/321127.321128`.

B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus. Haptic identification of objects using a modular soft robotic gripper. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1698–1705, Sept 2015. doi: 10.1109/IROS.2015.7353596.

Nick Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behavior*, 6(2):325–368, 1997. doi: 10.1177/105971239700600205. URL `https://doi.org/10.1177/105971239700600205`.

Nick Jakobi. Minimal simulations for evolutionary robotics. 1998.

Hod Lipson and Jordan B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.

Richard McDonnell. Helping robots get a grip on delicate products, 2015. URL `http://blog.parker.com/helping-robots-get-a-grip-on-delicate-products`. [Online; accessed 17-July-2018].

Gareth Monkman. Precise piezoelectric prehension. *Industrial Robot: An International Journal*, 27(3):189–194, 2000. doi: 10.1108/01439910010371605.

Giovanni Rateni, Matteo Cianchetti, Gastone Ciuti, Arianna Menciassi, and Cecilia Laschi. Design and development of a soft robotic gripper for manipulation in minimally invasive surgery: a proof of concept. *Meccanica*, 50(11): 2855–2863, Nov 2015. ISSN 1572-9648. doi: 10.1007/s11012-015-0261-6. URL `https://doi.org/10.1007/s11012-015-0261-6`.

H. Silva, S. M. Oliveira, and A. L. Christensen. Conillon: A lightweight distributed computing platform for desktop grids. In *6th Iberian Conference on Information Systems and Technologies (CISTI 2011)*, pages 1–6, June 2011.

Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 15–22, New York, NY, USA, 1994. ACM. ISBN 0-89791-667-0. doi: 10.1145/192161.192167. URL `http://doi.acm.org/10.1145/192161.192167`.

Russell Smith. Open dynamics engine, 2000. URL `http://www.ode.org/`.

Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99–127, June 2002. ISSN 1063-6560. doi: 10.1162/106365602320169811. URL `http://dx.doi.org/10.1162/106365602320169811`.

Kevin Tai, Abdul-Rahman El-Sayed, Mohammadali Shahriari, Mohammad Biglarbegian, and Shohel Mahmud. State of the art robotic grippers and applications. *Robotics*, 5(2), 2016.