MATERIALS AND MANUFACTURING PROCESSES

Morphological Evolution for Inspecting Fluid Environments Using Robot Operating System (ROS)

Ahmed Hallawa[a], Giovanni Iacca[b], Cagatay Sariman[a], Touhidur Rahman[a], Michael Cochez[c,d], and Gerd Ascheid[a]

[a] RWTH Aachen University, Kopernikusstraße 16, 52074 Aachen, Germany;
[b] University of Trento, Via Sommarive 9, 38123 Trento, Italy;
[c] Fraunhofer Institute for Applied Information Technology FIT, 52056 Aachen, Germany;
[d] Vrije Universiteit Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands

ARTICLE HISTORY
Compiled March 18, 2020

ABSTRACT
In many manufacturing processes, sensor agents specifically adapt to explore pipes and other constrained environments filled with fluid are usually needed for monitoring purposes. However, in some of these environments only miniaturized agents can be used. Furthermore, these agents might be kinetically passive, due to limited resources and size. Therefore, designing and using these agents can be difficult. One possible solution to this problem is to change the agents' morphology, such that optimally-shaped agents reach target destinations simply by passively moving through the fluid. Here, we propose an evolutionary scheme for evolving the agent's morphology to reach a predefined desired point in a fluid environment. This scheme includes a genotype-phenotype mapping based on Lindenmayer-Systems, as well as custom reproduction operators, selection criterion, and fitness function. In order to allow the simulation of irregularly-shaped bodies underwater, we develop a simulation framework based on the Robot Operating System and the Unmanned Underwater Vehicle package. We test the proposed method on a set of 10 target points in a pipe inspection scenario. Results show that the evolved agents reach the target points with a distance error smaller than 5% in the worst case, and a standard deviation of 1.1% over 10 repeated experiments.

KEYWORDS
Morphological, Evolution, Pipe, Inspection, L-Systems, ROS, UUV, Simulator, Conical, Frustum, Fluid, Simulation.

1. Introduction

In many manufacturing applications, such as in metallurgy plants, various kinds of water, oil, and gas distribution systems are used. In these systems, typically continuous monitoring of the status of pipes is needed to prevent leaks and other kinds of malfunctioning that might completely affect or disrupt the productivity of the plant.[1] In this regard, previous literature has already proposed various robotic systems for monitoring and inspection processes.[2–5] However, most of these robots are bulky, heavy, and expensive, and may require an interruption of the piping operation either partially or completely in order to perform the inspection task. This makes their use extremely

CONTACT Ahmed Hallawa. Email: hallawa@ice.rwth-aachen.de

costly, or even impossible in critical operations. Furthermore, in many applications pipe systems are difficult to access and not wide enough for these robots to function adequately.

An alternative solution is to use centimeter-sized sensor agent(s), which are able to inspect the pipes while they are still in operation. However, due to their scaled-down size these agents generally are *kinetically passive* and as such controlling their motion behaviour is challenging.[6] Furthermore, the environments to be inspected are typically GPS-denied (being the pipes underground, or shielded from external signals), which introduces further hurdles for localization and mapping.[7] One possible solution is to design the morphological properties of the agents in a way that would meet some predefined functional requirements, such as reaching a certain position in the environment at a given time. However, this optimization process is extremely complex as it is difficult to derive a mathematical representation of the problem, which involves hard-to-model fluid dynamics. In addition, it is expected that for a single functional objective, such as "spreading" the agents in the environment, multiple agents are needed and each of them might require a different morphological design. Therefore, black-box optimization techniques such as Evolutionary Algorithms (EAs) can offer a solution to this problem.[8] For example, EAs have been used in self-organizing assembly and adaptive multi-robot systems.[9–11] As for morphological evolution, EAs have been adopted in a wide range of robotics applications[12–17] but, to the best of our knowledge, the use of morphological evolution for passive miniaturized agents specifically meant for fluid environments has never been explored so far.

In this work, we present a methodology to evolve the morphological properties of an agent to achieve a given behavioural objective in a physics-based simulation of a pipe inspection scenario. Furthermore, in order to simulate the evolved agents in an efficient way, we extend an existing simulation package dubbed as Unmanned Underwater Vehicles (UUV).[18] This package is integrated with the widely adopted Robot Operating System (ROS)[1]. The combination of UUV and ROS allows the simulation of irregularly-shaped bodies, such as those we assume in the agents evolved with our framework, which are formed by a set of modules in the shape of a conical frustum.

To test our proposed framework, we set as objective of the evolutionary process the design of agents capable of reaching a set of predefined points in a fluid environment, starting from an insertion point in the pipe. It is important to highlight that in the tested scenario the evolved agents reach their target point relying only on their morphology, without any form of actuation or active interaction with the environment. As such, these agents are extremely cost-effective and can be easily deployed in real-world environments.

To summarize, the main contributions -and the relative challenges- of this work are:

- We hypothesize that the control of sensor agents can be attained by evolving *only* their morphology. This represents a radical departure from the existing works on morphological evolution for robotics, where usually motion control is obtained by evolving both the morphology *and* the controller. As such, our problem is much more challenging since assuming that agents are passive reduces the degrees of freedom (due to the lack of actuation) and therefore optimizing their morphology is much more critical than the cases where actuation is possible. As we show in our experiments, very different morphologies are needed to reach target points that are even very close to each other. This means that the problem (and the search for an optimal solution) is very sensitive to the target, and suboptimal

[1]Available at `http://www.ros.org`.

solutions might not solve the task at all.

- We provide a complete evolutionary-driven simulation framework that couples an EA with physics-based simulations based on ROS and UUV. The EA is capable of exploring a large morphological space consisting of thousands of possible variable-length (in terms of number of modules) morphologies. Each morphology is then simulated with a high accuracy and as such can be eventually 3D-printed and tested in the real world. This proposed combination of ROS and UUV (and the physical model therein) is novel and represents a major contribution w.r.t. the existing literature.
- We show a proof-of-concept application of the proposed framework on a pipe inspection scenario.

The remaining of the paper is structured as follows. The next section discusses the characteristics of the agents considered in our scenario, the details of the Evolutionary Algorithm (genotype-phenotype mapping, reproduction and selection), and the details of the extension to the UUV simulator. Then, section 3 presents the numerical results. Finally, we give the conclusions in section 4.

2. Materials and methods

2.1. Morphological evolution

As discussed in the previous section, our methodology for evolving the sensor agents' morphology is based on the use of Evolutionary Computation. Following the typical loop of an Evolutionary Algorithm, we start with initializing a population of random solutions, in our case a population of agents with different morphology. Afterwards, an evaluation based on a fitness function is executed on each agent. This requires simulating the interaction of the agent with the environment, which presents a challenge as such simulation is usually either too slow or too inaccurate. Section 2.4 is dedicated to this aspect.

Agents' evaluations are followed by fitness-proportionate selection, which picks the agents that will "reproduce", i.e. generate new solutions that will partially inherit the parents' traits (in our case, morphological traits). Reproduction occurs by means of custom crossover and mutation operators, that act both on the genotype of the agents. The latter identifies the set of the relevant geometrical parameters that affect the morphology of the agent, thus its phenotype. In our case, the genotype-phenotype mapping is based on an indirect genetic representation that uses a Lindenmayer-System (L-System) [19], as discussed below.

2.2. Genotype-phenotype mapping

The representation of the genotype is an important factor in the evolutionary process. Here, we do not use a direct binary encoding to represent the morphology of the agent. Instead, our representation of the genotype is indirect, based on a mechanism called Lindenmayer-System. According to this mechanism, each agent is a morphological realization of its grammar rules. The grammar of such L-System is defined as a triple $\mathcal{G} = (A, w, P)$, where:

- A is an alphabet, i.e. a set of symbols containing replace, replaceable and non-replaceable elements;

- w is an axiom, i.e. a symbol from which the system starts;
- P is a set of production rules, which take as an input a replaceable symbol and replace it with a fixed sequence of symbols from V.

The system generates a string by starting from the axiom w and then repeatedly applying one of the production rules for a fixed number of steps. In our design, the evolved agents consists of n modules, where n is not fixed but rather is subject to the evolutionary process. I.e., the length of the agent is evolved. Each module is a hollow conical frustum realized as a solid of revolution, i.e. a solid figure obtained by rotating a curve around an axis. In our case, the curve is a line defined by its length and the angle of contact relative to one of its neighboring modules. To represent these properties in the genotype, we set in the production rules: (1) a set of commands to allow adding modules in both directions along the main axis of the evolved shape; (2) a set of tilting commands to change the angle of contact of the new module; and (3) a set of possible angles and lengths. Consequently, the adopted L-System is defined as follows:

- Each production rule in P contains 3 quintuples.
- A quintuple consists of 5 letters: an "add" command, a "tilt" command, a degree, a block type and a move operand.
- The initial rule has an extra 'C' at the beginning, to ensure that the core module (the part of the agent that contains its hardware controller) is part of the evolved agent. I.e., the axiom is 'C'.
- Three iterations are performed to generate the genome.

In our scheme, we assumed that the user has at least one shape that must be enforced as a constraint (the core module, represented by the axiom 'C'). This is because in many cases the developed hardware controller is designed to fit a particular volume with predefined specifications. Table 1 summarizes the alphabet of the designed L-System and the meaning of each symbol. It is worth mentioning that in case of repetition of a certain action in the genome, such as "add" or "tilt", only one is applied and the others are discarded.

Figure 2 shows an example of the phenotype realization of an evolved agent consisting of three module in addition to the core module. In this example, a core module (a sphere) is already present, thus the reference point corresponds to the core module itself. Let us try to read the following genotype:

```
addB-tiltn-1-M-moveB-addB-tiltn-2-S-moveB-addB-tiltp-1-L-moveB
```

The first `addB` command adds a new module backwards relative to the reference point at the core module, then `tiltn` forces the angle to the module to be calculated clockwise. A value of 1 sets the tilt angle to be 10, afterwards, the length of the module is given to be medium (defined by the 'M' symbol) and finally `moveb` will move the reference point one step back to the new module. Following the same rules, one can build the three modules, as shown in Figure 2 (top). This can be turned into a 3D shape by simply rotating the figure with a preset thickness around the center axis (solid of revolution), as shown in Figure 2 (bottom).

2.3. Reproduction and selection

Two possible reproduction operations are possible in the presented scheme: crossover and mutation. Crossover is conducted by randomly picking a single point among the rules in the genotype, and exchange the gene sequence around this random point between two parent solutions, thus producing two new offspring. Mutation is executed on offspring (after crossover) by flipping any symbol in their genotype, e.g. flipping an `addF` to `addB`, or a `moveA` to `moveB`. Both crossover and mutation are executed with probability $P_{crossover}$ and $P_{mutation}$, which are two hyper-parameters to be selected adequately to allow a proper exploration-exploitation balance. As for selection, we adopted a roulette-wheel (fitness-proportionate) selection criterion, where the agents with better performance are more likely to reproduce. Furthermore, we kept the population size constant and did not use any elitism.

2.4. Simulator

As explained earlier, one important aspect of the proposed evolutionary scheme is the evaluation of the candidate solutions. To achieve this, we defined an evolutionary-driven simulation framework that can be summarized as in Figure 1. In this framework, the EA loop starts with the initialization of the population of agents. To realize an agent, the open source application openSCAD[2] is used. The output of openSCAD is a 3D model of the agent, which can be used by ROS in order to evaluate it. In this regard, the ROS package UUV is adopted to instantiate the agent and the fluid environment in order to conduct the simulation. After the simulation, the fitness is calculated using the output from the agent and the environment, based on the fitness definition. For example, if the agent fitness is defined by the velocity of the agent, then the details of the covered distance and the time needed to cover this distance by the agent are saved during the simulation and used afterwards by the EA loop to calculate the fitness value. After evaluation, selection and reproduction are executed, which are followed by the evaluation of the new generation. This is repeated until the stopping criterion is achieved. However, the evolutionary process introduced before produces irregularly-shaped bodies (formed by a set of modules in the shape of a conical frustum), which are not possible to simulate using the current version of the UUV package. Therefore, in order to evaluate (simulate) the behavior of our evolved agents, we need to extend the UUV package. The reason for using and extending this package is mainly its ability to simulate a wide range of sensors, such as pollution sensors, in addition to pollution plumes. Furthermore, UUV can be easily integrated with ROS and Gazebo, which offers a wide range of libraries essential for simulating multi-robot and/or multi-agent systems.

We now give an overview of our extension to the UUV package. The objective is to be able to simulate objects evolved using the evolution scheme presented in the previous sections. What we expect as output of the evolutionary scheme is an agent with multiple modules, each with the shape of a conical frustum, as shown in Figure 3. To simulate the interaction of the slender body of the evolved agents with the environment, the following attributes need to be calculated for each module: added mass coefficients, moments of inertia (MOI), linear and quadratic damping coefficients, center of gravity (COG), center of buoyancy (COB), and finally total mass and volume. For simplicity, we assume that there is no damping, thus linear and quadratic damping coefficients are set to 0. In the following, we discuss the other attributes and how they are computed.

[2]Available at `https://www.openscad.org`.

2.5. Mass coefficients

Strip theory is suited for calculating the mass coefficients for slender bodies, as it is a widely adopted and sufficiently accurate method.[20] However, by using this method we cannot find the added mass coefficient in the direction of the X-axis, as shown in Figure 3. For this direction, we adopt the equivalent ellipsoid method. This method finds 3D mass coefficients M_{ij}, where M_{ij} stands for the 3D added mass coefficient in the i^{th} direction due to a unit acceleration in the j^{th} direction. Similarly, $m_{kl}(x)$ indicates the added mass in the k^{th} direction of a 2D cross section at location x due to a unit acceleration in the l^{th} direction. L_1 and L_2 represent the coordinates of the agent's two ending points along the x_1-axis, while the origin is located at the agent's center of buoyancy (COB). Thus, based on the strip theory, the following formulas hold:

$$M_{22} = \int_{L_1}^{L_2} m_{22}(x)dx \tag{1}$$

$$M_{35} = -\int_{L_1}^{L_2} x\, m_{33}(x)dx \tag{4}$$

$$M_{33} = \int_{L_1}^{L_2} m_{33}(x)dx \tag{2}$$

$$M_{55} = \int_{L_1}^{L_2} x^2\, m_{33}(x)dx \tag{5}$$

$$M_{26} = \int_{L_1}^{L_2} x\, m_{22}(x)dx \tag{3}$$

$$M_{66} = \int_{L_1}^{L_2} x^2\, m_{22}(x)dx \tag{6}$$

In this work, all evolved agents have rotationally symmetrical bodies with respect to x_1-axis. Therefore, the added mass matrix simplifies to:

$$\begin{bmatrix} M_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & M_{22} & 0 & 0 & 0 & M_{26} \\ 0 & 0 & M_{33} & 0 & M_{35} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & M_{35} & 0 & M_{55} & 0 \\ 0 & M_{26} & 0 & 0 & 0 & M_{66} \end{bmatrix}$$

with $M_{22} = M_{33}$; $M_{26} = -M_{35}$ and $M_{55} = M_{66}$.

On the other hand, since forces in the x_1-direction cannot be determined with strip theory as mentioned earlier, another method is needed to calculate the remaining 3D added mass coefficient M_{11}. Here we adopt the method of the equivalent ellipsoid, by assuming that the evolved agents can be modelled as an elongated ellipsoid. Thus, based on theory of hydrostatics, M_{11} is given as follows:

$$M_{11} = \frac{4}{3}\pi a b^2 \rho\, k_{11} \tag{7}$$

where ρ is the density and k_{11} is the hydrodynamic coefficient, defined as:

$$k_{11} = \frac{\alpha_0}{2 - \alpha_0} \tag{8}$$

where:

$$\alpha_0 = \frac{2(1-\beta^2)}{\beta^3}\left[\frac{1}{2}ln\left(\frac{1+\beta}{1-\beta}\right)-\beta\right] \quad (9) \qquad \beta = \sqrt{1-\frac{b^2}{a^2}} = \sqrt{1-\frac{d^2}{L^2}} \quad (10)$$

where d and L are the maximum diameter and length for the whole object respectively.

2.6. Moment of inertia (MOI)

Hollow conical frustum is a solid of revolution, i.e. a solid figure obtained by rotating a plane curve around an axis. Solids of revolution are defined by the generator functions $f_1(x)$ and $f_2(x)$, where $f_1(x)$ is the inner function of the solid of revolution, while $f_2(x)$ is the outer function, thus $0 \leq f_1(x) \leq f_2(x)$ must hold. What is special in our case is that our conical frustum is obtained by rotating an inclined rectangle plane around the X-axis. Thus $f_1(x)$ and $f_2(x)$ are simply two straight line equations.

In order to find the moment of inertia with respect to the X-axis, the total moment of inertia in its general form can be formulated as follows[21]:

$$I_X = \int_{x_0}^{x_f}\left\{\int_{f_1(x)}^{f_2(x)}\left[\int_{\theta_0}^{\theta_f}\rho(x,r_x,\theta)d\theta\right]r_x^3 dr_x\right\}dx \tag{11}$$

This expression can be used for any solid of revolution. For a complete revolution of the solid with constant density $\rho = \rho(x, r_x)$ the formula can be simplified as:

$$I_X = 2\pi\int_{x_0}^{x_f}\int_{f_1(x)}^{f_2(x)}\left[\rho(x,r_x)r_x^3 dr_x\right]dx \tag{12}$$

and simplified further for $\rho = \rho(x)$:

$$I_X = \frac{\pi}{2}\int_{x_0}^{x_f}\rho(x)\left[\underbrace{f_2(x)^4 - f_1(x)^4}_{M_{dx}}\right]dx \tag{13}$$

As shown in Equation 13, for solids of revolution with a density depending just on x (the height), and for homogeneous bodies, the volume integral is reduced to a simple integral in one variable. The only required parts are the functions that generate the solid and the limits along the X-axis. In our case, both $f_i(x)$ and $f_i(x)$ are equations of a straight line with the same slope m. Thus, the mechanical differential M_{dx} can be found as:

$$\begin{aligned} f_i(x) &= m(x-a)+c_i \\ f_i(x)^2 &= m^2(x-a)^2 + 2mc_i(x-a)+c_i^2 \\ f_i(x)^4 &= m^4(x-a)^4 + 4m^3c_i(x-a)^3 + 6m^2c_i^2(x-a)^2 + 4mc_i^3(x-a) + c_i^4 \end{aligned}$$

As a result, M_d can be defined as follows:

$$\begin{aligned} M_{dx} &= f_2(x)^4 - f_1(x)^4 \\ &= 4m^3(c_2-c_1)(x-a)^3 + 6m^2(c_2^2-c_1^2)(x-a)^2 + 4m(c_2^3-c_1^3)(x-a) + (c_2^4-c_1^4) \end{aligned}$$

Therefore:

$$
\begin{aligned}
I_X = \frac{\pi}{2}\,\rho \int_{x_0}^{x_f} M_{dx} dx = \frac{\pi}{2}\rho m^3(c_2-c_1)(x-a)^4 + 2m^2(c_2^2-c_1^2)(x-a)^3 \\
+ 2m(c_2^3-c_1^3)(x-a)^2 + (c_2^4-c_1^4)x\Bigg|_{x_0}^{x_f}
\end{aligned}
$$

where x_0 and x_f are the start and end of each module relative to the center of mass of the overall shape.

As for the moment of inertia with respect to the Y and Z axes, we need to consider that all the solids evolved in this work are generated through a complete revolution and possess a constant density. Therefore, they exhibit cylindrical symmetry. That means that their moment of inertia with respect to the Z-axis is always equal to their moment of inertia with respect to the Y-axis. Thus:

$$I_Z = I_Y \tag{14}$$

The moment of inertia in its general form for the Y-axis (and Z-axis) is given by:

$$
\begin{aligned}
I_Y = \int_{x_0}^{x_f} \left\{ \int_{f_1(x)}^{f_2(x)} \left[\int_{\theta_0}^{\theta_f} \rho(x,r_x,\theta)\sin^2\theta d\theta \right] r_x^3 dr_x \right\} dx \\
+ \int_{x_0}^{x_f} \left\{ \int_{f_1(x)}^{f_2(x)} \left[\int_{\theta_0}^{\theta_f} \rho(x,r_x,\theta) d\theta \right] r_x dr_x \right\} x^2 dx
\end{aligned}
\tag{15}
$$

For a complete revolution with $\rho = \rho(x, r_x)$ and I_X as in Equation 13, this formula can be simplified as:

$$I_Y = \frac{1}{2}I_X + 2\pi \int_{x_0}^{x_f} x^2 \int_{f_1(x)}^{f_2(x)} \left[\rho(x,r_x) r_x dr_x \right] dx \tag{16}$$

Finally, for $\rho = \rho(x)$, I_Y can be further reduced to:

$$I_Y = \frac{1}{2}I_X + \pi \int_{x_0}^{x_f} \rho(x)\, \underbrace{x^2\left[f_2(x)^2 - f_1(x)^2\right]}_{M_{dy}} dx \tag{17}$$

Generally, this formula holds:

$$
\begin{aligned}
x^2 f_i(x)^2 &= x^2m^2(x-a)^2 + 2mc_i x^2(x-a) + c_i^2x^2 \\
&= x^2m^2(x-a)^2 + 2mc_i x^3 - 2mac_i x^2 + c_i^2x^2
\end{aligned}
$$

Therefore, similar to M_{dx}, M_{dy} is calculated as follows:

$$
\begin{aligned}
M_{dy} &= x^2f_2(x)^4 - x^2f_1(x)^4 \\
&= 2m(c_2-c_1)x^3 - 2ma(c_2-c_1)x^2 + (c_2^2-c_1^2)x^2
\end{aligned}
$$

Finally:

$$I_Y = \frac{1}{2}I_X + \pi \int_{x_0}^{x_f} M_{dy}dx = \frac{1}{2}m(c_2 - c_1)x^4 - \frac{2}{3}ma(c_2 - c_1)x^3 + \frac{1}{3}(c_2^2 - c_1^2)x^3 \Big|_{x_0}^{x_f}$$

As mentioned earlier, this formula (based on Equation 17) can be adopted for the moment of inertia Z-axis as well.

2.7. Center of gravity (COG) and center of buoyancy (COB)

Center of gravity and center of buoyancy are paramount in order to model the evolved shapes adequately. Firstly, for the hollow conical frustum the center of gravity (with a constant wall thickness) can be calculated as follows:

$$COG = x_0 + \frac{h * (R_b + 2R_t)}{3 * (R_b + R_t)} \tag{18}$$

where R_b is the bottom radius, until the middle point of the wall, R_t is the top radius, until the middle point of the wall, and h is the height. On the other hand, the center of gravity of a solid conical frustum (to be used as center of buoyancy) can be calculated as follows:

$$COG = x_0 + \frac{h}{4} * \frac{R_b^2 + 2R_bR_t + 3R_t^2}{R_b^2 + R_bR_t + R_t^2} \tag{19}$$

where $R_b = 0$ is the bottom radius until the middle point of the wall, R_t is the top radius until the middle point of the wall, and h is the height.

The global center of gravity for the agent can be computed from the centers of each of the n modules as follows:

$$COG = \frac{\sum_{i=1}^{n} COG_i * Mass_i}{Mass_{agent}} \tag{20}$$

where COG_i is the center of gravity of module i (relative to the origin), $Mass_i$ is the mass of module i, and $Mass_{agent}$ is total mass of the agent.

2.8. Volume and mass

Finally, we need to calculate the volume, mass and density of the shape. For the solid conical frustum, the volume can be calculated as follows:

$$V = \frac{1}{3}\pi h(R_b^2 + R_bR_t + R_t^2) \tag{21}$$

For the sphere:

$$V = \frac{4}{3}\pi R^3 \tag{22}$$

For the spherical cap:

$$V = \frac{1}{3}\pi R^3(2 - 3sin\theta + sin^3\theta) \tag{23}$$

Finally, the mass of all these shapes can be calculated as follows:

$$m = V\rho \tag{24}$$

3. Results and discussion

To test our framework, we use a case study where the objective is to design agents capable of reaching a set of predefined target points in a tube filled with flowing water. A graphical representation of the simulated scenario is shown in Figure 4. The environment conditions are set as follows. The length of the pipe is 9 meters, and its diameter is 2 m. The water level almost reaches the upper part of the pipe. The water current has a velocity of $1\,\mathrm{ms^{-1}}$, and the density of the water is $1024\,\mathrm{kgm^{-3}}$.

Se define 10 target points in various parts of the pipe, such that a different motion behavior with respect to longitudinal/angular velocity and acceleration (and thus, a resulting trajectory) is needed in order to reach each of these target points. These points are chosen to cover a wide range of cases, including cases where that agent requires to ascend in the pipe. Since we assume *passive* agents (without actuation), their trajectory will only be determined by their morphology. Therefore, each agent needs to be optimally shaped for each target point, such that if it is injected into the pipe at a fixed position (insertion point), it will reach the given target point.

As discussed earlier, each evolved agent must include a spherical control unit (the core module that hosts the agent's hardware and sensors). This constraint is added to minimize the reality gap, as in many applications it is required to have a specific shape included in the overall agent structure. In this regard, we set the mass of the core module to 0.05 kg and its radius to 0.01 m. In addition, the following characteristics of the agent are defined:

(1) Length of the small module = 0.0025 m
(2) Length of the medium module = 0.005 m
(3) Length of the large module = 0.01 m
(4) Thickness of the shell = 0.001 25 m
(5) Density of the material (Acrylonitrile Butadiene Styrene[3]) = $1050\,\mathrm{kgm^{-3}}$

It is worth mentioning that the number of iterations of the L-system is 3, such that the maximum number of modules is 40. Consequently, the highest and lowest possible length for an evolved agent is 0.4 m (39 modules * 0.01 m + 0.01 core radius) and 0.1075 m (39 modules × 0.0025 m + 0.01 m core radius)

To solve this test problem, we use the evolutionary scheme and the simulation setup presented in the previous sections. We set the fitness of an individual agent (to be maximized) as follows:

$$fitness = 100 \times \left(1 - \frac{d_{min}}{d_{max}}\right) \tag{25}$$

[3]It is worth mentioning that adopting the density of Acrylonitrile Butadiene Styrene (ABS) in the evolution process and using openSCAD facilitates the realization of the evolved agents using 3D printers.

where d_{min} is the smallest Euclidean distance between the simulated trajectory of the agent and the target point, and d_{max} is the distance between the furthest possible position in the environment and the target point. As such, we express the fitness as a percentage (the optimal value 100% is obtained when $d_{min} = 0$).

Furthermore, we set the population size to 10 and the number of generations to 5, i.e. in total we run 50 evaluations per evolutionary run. The crossover and mutation probabilities, $P_{crossover}$ and $P_{mutation}$, are set to 0.9 and 0.1 respectively.

Due to the stochastic nature of the evolutionary process, for each target point we repeat the evolutionary algorithm 10 times. All the evolutionary experiments were conducted on a Linux PC with Intel(R) Xeon(R) CPU W3540 @ 2.93GHz and 24 GB RAM. The total computational time for 5000 evaluations (50 evaluations per evolutionary run, $\times$ 10 target points $\times$ 10 repetitions) is 20 hours.

Figure 5 shows the fitness trends in terms of mean fitness and the respective standard deviation at every generation across the 10 repeated evolutionary runs for 8 of the 10 target points. It can be seen from the figure that the evolved agents are able to reach the target points with a relative distance error that is smaller than 5% in the worst case. Furthermore, the different runs display a small standard deviation (1.1% across 10 runs), thus showing the strong robustness of the proposed method.

In Figure 6, the best evolved agent's shape obtained across the 10 different repetitions for each of the 10 target points is shown. Of note, in many cases the morphological properties of the evolved agents for two adjacent target points (see e.g. TP 6 and TP7) are significantly different. This confirm our hypothesis that different morphologies are really needed to reach different target points: as a consequence, a single morphology cannot reach all the target points. We further hypothesize that this is due to the complexity of the morphological space. For example, a small change in the height of the target points (w.r.t. the bottom of the pipe) may require a change in the center of buoyancy of the agent, such that in order to reach two adjacent targets a change in the morphological properties is needed.

Finally, we have performed a set of experiments aimed at testing the consistency (replicability) of the simulator. In order to do that, we have re-simulated for 10 times each best performing agent obtained for each target point, and recorded the relative trajectories. Figure 7 shows the mean trajectory followed by the best agents related to 8 of the 10 target points, in addition to the standard deviation of the position of the agent along the Y-axis w.r.t. the X-axis. It should be noted that in the presented plots, once the agent has reached the minimum distance to its target point, the trajectory is not recorded further. Once again, it can be seen from this figure that the trajectories followed by the agent in each experimental condition are quite robust across multiple runs, as the standard deviation across the 10 repeated trajectories is smaller than 1% in the worst case. Furthermore, we note that different morphologies follow completely different trajectories, which are highly optimized in order to reach each target point: this further confirms that a single morphology would not be able to reach all the target points.

4. Conclusions

In this paper, we presented an evolutionary scheme to evolve scaled-down sized agents for inspection and exploration of fluid environments. The proposed evolutionary scheme includes an indirect genotype-phenotype mapping based on Lindenmayer-Systems, according to which each evolved agent is constructed from a set of modules with the shape of a conical frustum. This scheme allows an adequate exploration of a large morphological space by means of a set of production rules. Furthermore, it allows setting constraints on the upper and lower limits of several physical features of each module, such as its length and radius. In addition, our scheme includes the possibility to integrate a predefined module, e.g. a control unit, to be inserted in the evolved agent. To test our scheme, we applied it on a problem where the objective was to obtain agents capable to reach 10 target points in a given pipe environment. In our experiments, the evolved agents, relying only on their morphology, were able to consistently reach the target points with a relative distance error smaller than 5% in the worst case, and a standard deviation of 1.1% across 10 repetitions of the evolutionary loop. Finally, to test the consistency of our simulator, we repeated the simulation of the best scoring agent for each point 10 times. Results show a standard deviation that is smaller than 1% in the worst case, highlighting a good replicability of the simulation w.r.t. the random seed.

The present work can be extended in various ways. In particular, we plan to test the proposed framework on more complex environments (for instance consisting of multiple pipes connected through junctions) and conduct real-world tests on a controlled experimental setup, in order to analyze the reality gap between the simulator and the real environment under investigation. Finally, it will be interesting to test the proposed framework on real industrial cases of monitoring of water and oil pipes.

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No: 665347.

References

[1] Srijith, B.; Thirumalai, R. Inspection tool for flexible manufacturing system. *Imp. J. Interdiscip. Res.* **2017**, *3*.

[2] Nishihara, T.; Osuka, K.; Tamura, I. Development of a simulation model for inner-gas-pipe inspection robot: SPRING. Proceedings of the SICE Annual Conference. 2010; pp 902–904.

[3] Kim, J.; Sharma, G.; Boudriga, N.; Iyengar, S. S. SPAMMS: A sensor-based pipeline autonomous monitoring and maintenance system. Proceedings of the International Conference on COMmunication Systems and NETworks (COMSNETS). 2010; pp 1–10.

[4] Nayak, A.; Pradhan, S. Design of a new in-pipe inspection robot. *Procedia Eng.* **2014**, *97*, 2081–2091.

[5] Aaqib, H. S.; Fazil, A. M.; Saquib, H. Pipe Inspection Robot. *Int. Res. J. Eng. Techn.* **2018**, *5*.

[6] Tur, M.; M., J.; Garthwaite, W. Robotic devices for water main in-pipe inspection: A survey. *J. Field Robot.* **2010**, *27*, 491–508.

[7] Hallawa, A.; Schlupkothen, S.; Iacca, G.; Ascheid, G. Energy-efficient environment mapping via evolutionary algorithm optimized multi-agent localization. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Companion. 2017; pp 1721–1726.

[8] Pasemann, F. structure and function of evolved neuro-controllers for autonomous robots. *Connect. Sci.* **2004**, *16*.

[9] Brezocnik, M.; Balic, J. A genetic-based approach to simulation of self-organizing assembly. *Robot. Comp.-Int. Manuf.* **2001**, *17*, 113–120.

[10] Meyers, R. A. *Encyclopedia of complexity and systems science*; Springer, 2009.

[11] Hrabia, C.-E.; Lützenberger, M.; Albayrak, S. Towards adaptive multi-robot systems: self-organization and self-adaptation. *Knowl. Eng. Rev.* **2018**, *33*.

[12] Brodbeck, L.; Hauser, S.; Iida, F. Morphological evolution of physical robots through model-free phenotype development. *PLoS ONE* **2015**, *10*, 1–17.

[13] Doncieux, S.; Mouret, J.-B.; Bredeche, N.; Padois, V. Evolutionary Robotics: Exploring New Horizons. In *New Horizons in Evolutionary Robotics*; Doncieux, S., Bredeche, N., Mouret, J.-B., Eds. 2011; pp 3–25.

[14] Floreano, D.; Nolfi, S. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*; Intelligent Robotics and Autonomous Agents series; MIT Press, 2004.

[15] Silva, F.; Correia, L.; Christensen, A. Evolutionary robotics. *Scholarpedia* **2016**, *11*, 33333.

[16] Jelisavcic, M.; de Carlo, M.; Hupkes, E.; Eustratiadis, P.; Orlowski, J.; Haasdijk, E.; Auerbach, J. E.; Eiben, A. E. Real-world evolution of robot morphologies: a proof of concept. *Artif. Life* **2017**, *23*, 206–235.

[17] Horodinca, M.; Doroftei, I.; Mignon, E.; Preumont, A. A simple architecture for in-pipe inspection robots. Proceedings of the International Colloquium on Autonomous and Mobile Systems. 2002; pp 61–64.

[18] Manhães, M. M. M.; Scherer, S. A.; Voss, M.; Douat, L. R.; Rauschenbach, T. UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation. Proceedings of OCEANS MTS/IEEE Monterey. 2016; pp 1–8.

[19] Hornby, G. S.; Pollack, J. B. Evolving L-Systems to generate virtual creatures. *Comput. Graph.* **2001**, *25*, 1041–1048.

[20] Sen, D. T.; Vinh, T. C. Determination of added mass and inertia moment of marine ships moving in 6 degrees of freedom. *Int. J. of Transp. Eng. Techn.* **2016**, *2*, 8–14.

[21] Diaz, R. A.; Herrera, W. J.; Martinez, R. Some powerful methods to calculate moments of inertia. *arXiv e-prints* **2004**, physics/0404005.

Figures and tables

Notation	Meaning
addF	Add the module forward relative to the current reference position
addB	Add the module backward relative to the current reference position
tiltp	Tilt positively
tiltn	Tilt negatively
0	Set tilting angle to 0
1	Set tilting angle to 10
2	Set tilting angle to 20
S	Set the length of the added module to small (e.g. 5cm)
M	Set the length of the added module to medium (e.g. 10cm)
L	Set the length of the added module to large (e.g. 15cm)
moveF	Move the reference point one module forward
moveB	Move the reference point one module backward
moveN	Do not move the reference point

Table 1. Alphabet A of the designed L-System and meaning of each symbol.

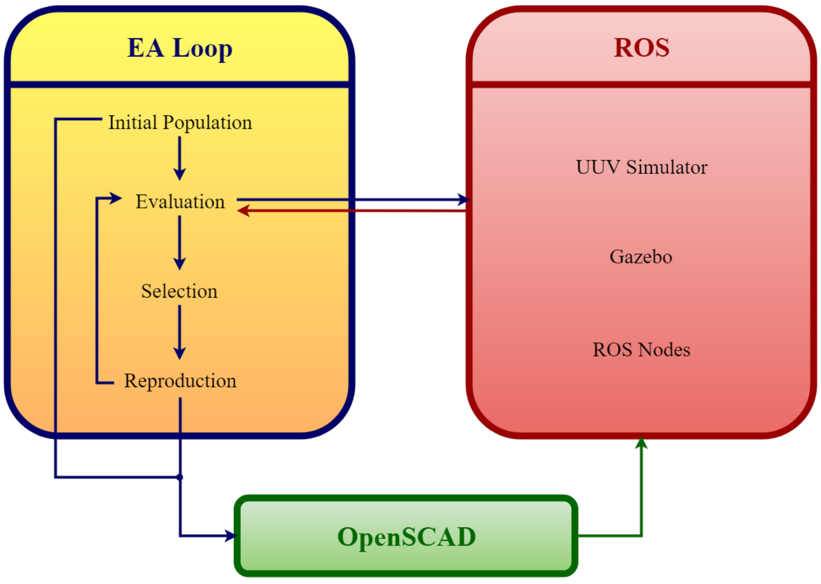


Figure 1. Framework overview.

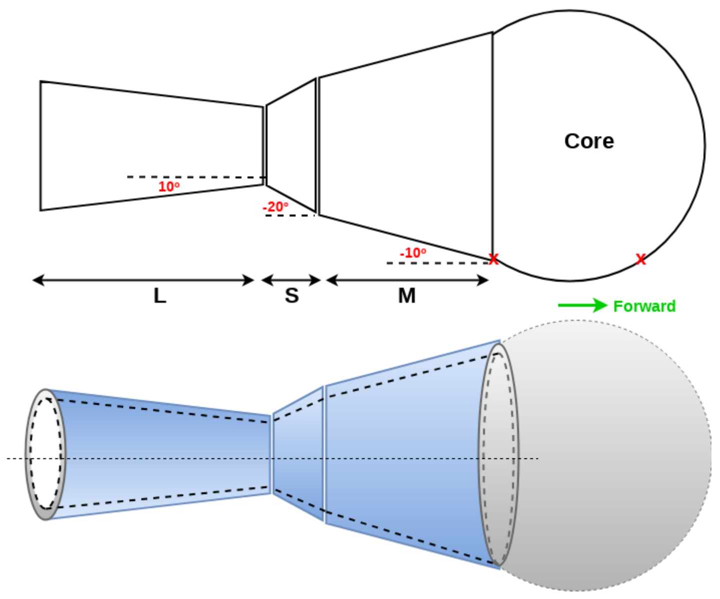


Figure 2. Phenotype realization: an example.

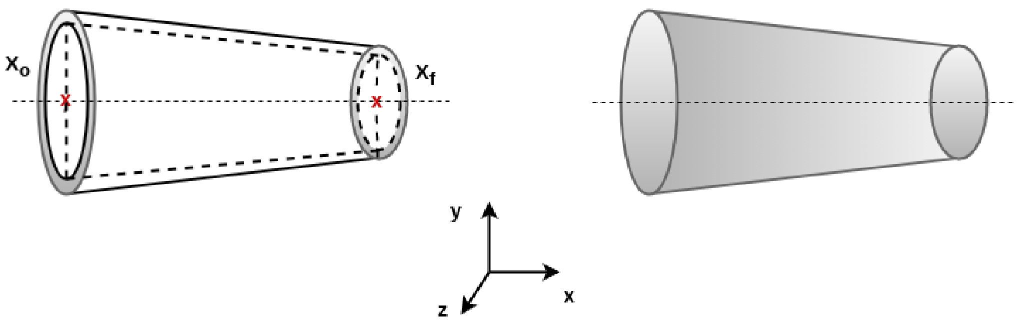


Figure 3. Conical frustum: hollow and solid.

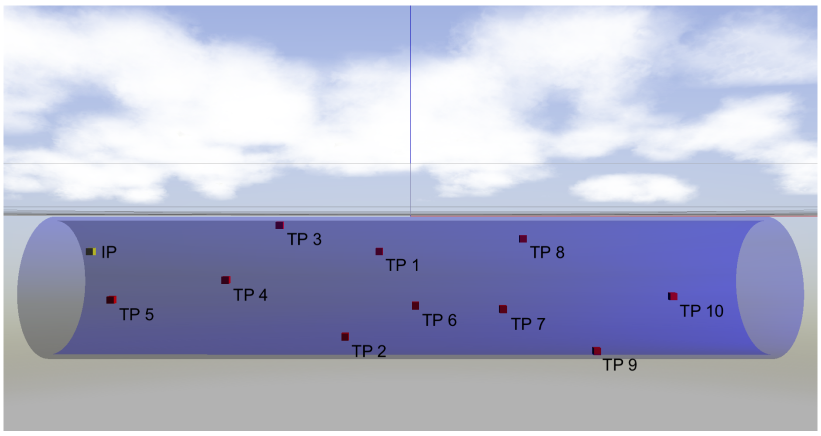
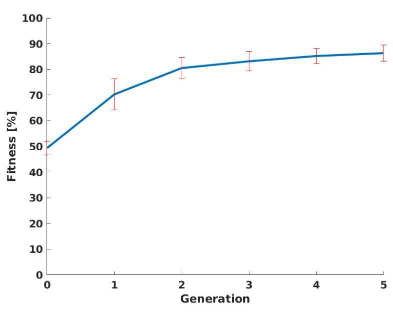
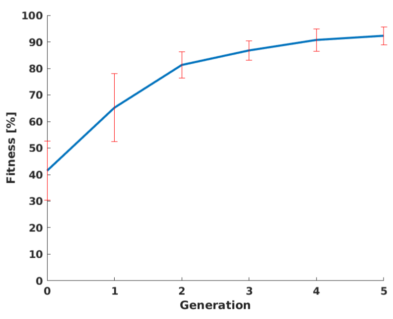


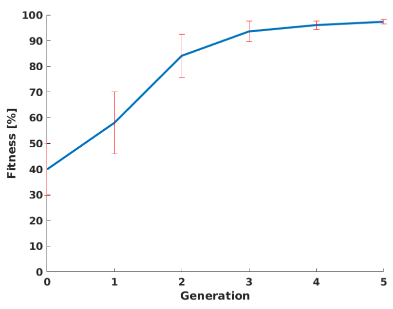
Figure 4. Simulated environment with insertion point ("IP") and target points ("TP").



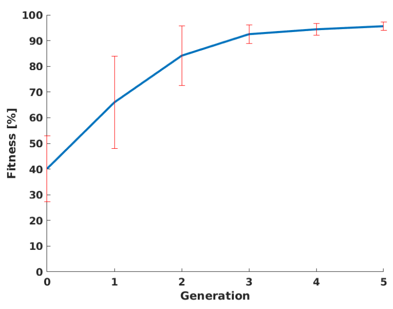
Target point 1



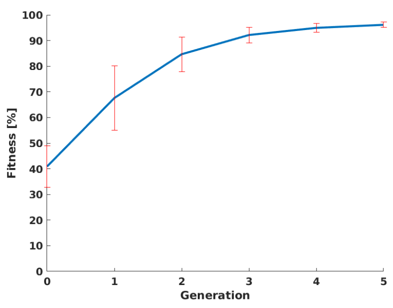
Target point 4



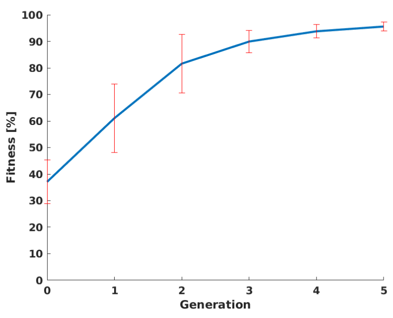
Target point 5



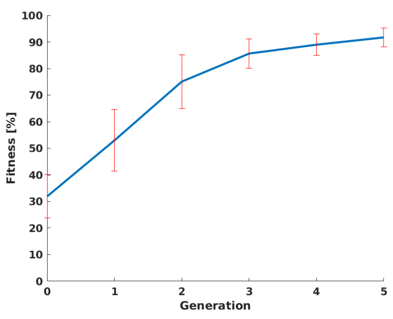
Target point 6



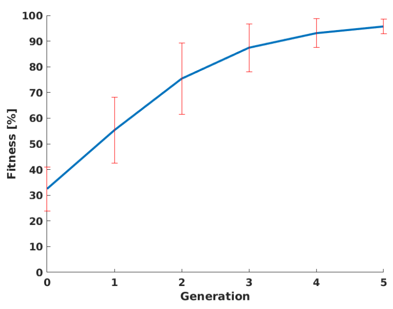
Target point 7



Target point 8



Target point 9



Target point 10

Figure 5. Fitness trend (mean and standard deviation at each generation, across 10 repetitions) of the evolutionary algorithm for 8 (out of 10) target points.

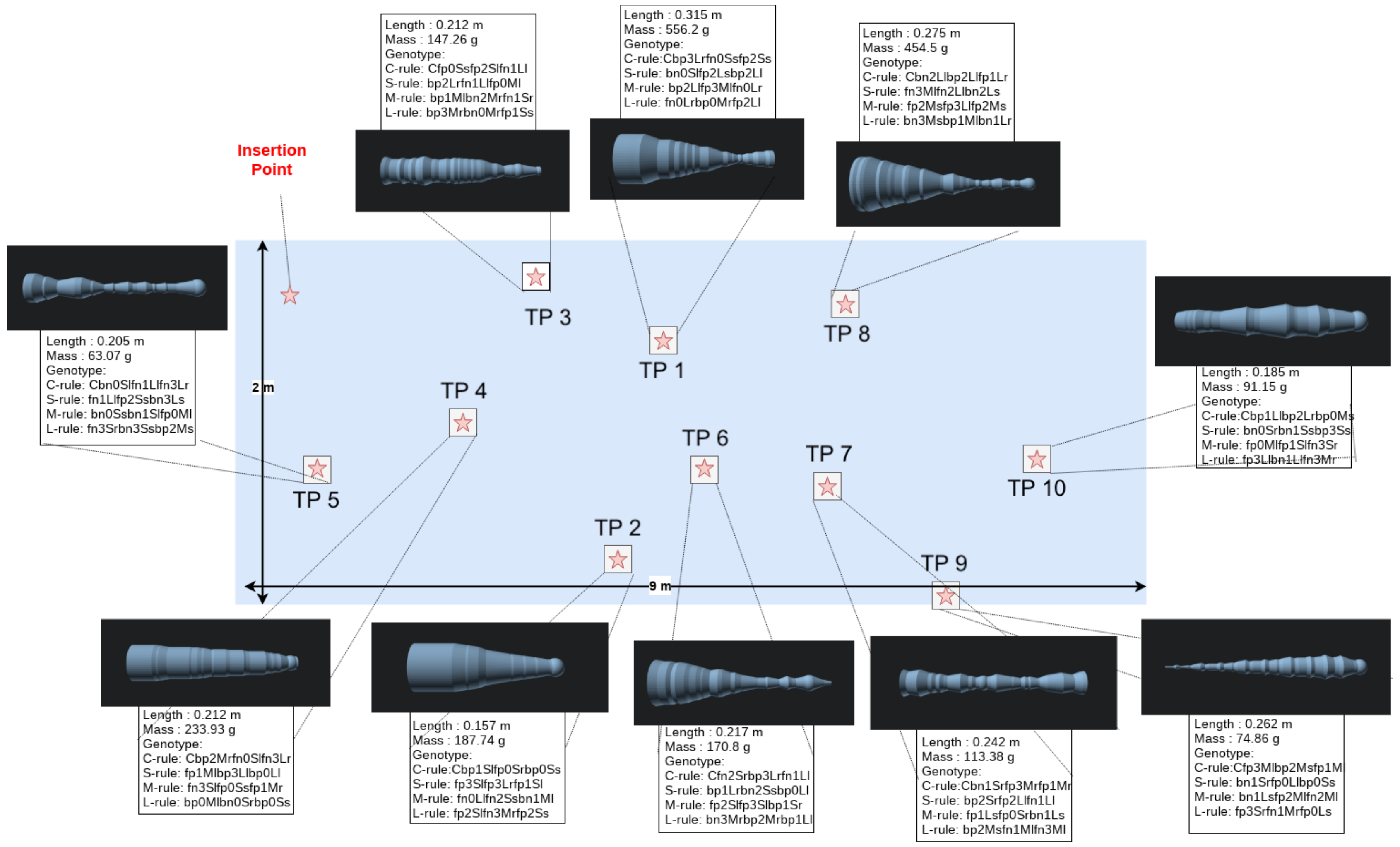
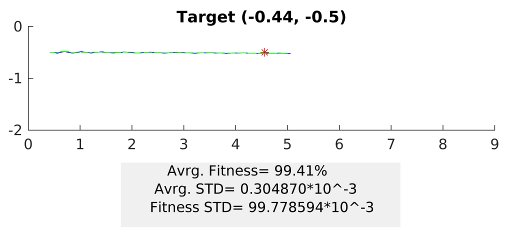
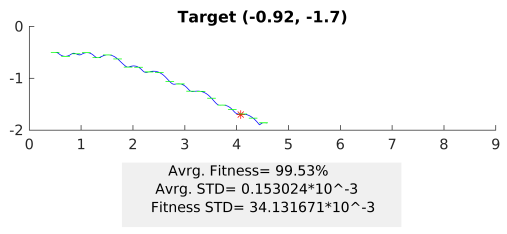


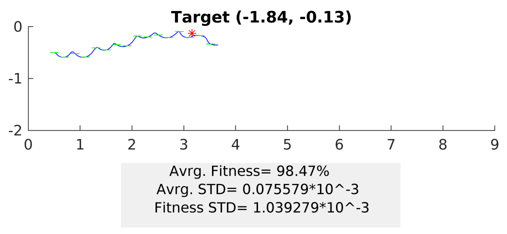
Figure 6. Best evolved agent for each of the 10 target points ("TP").



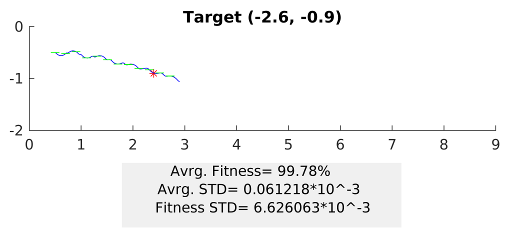
Target point 1



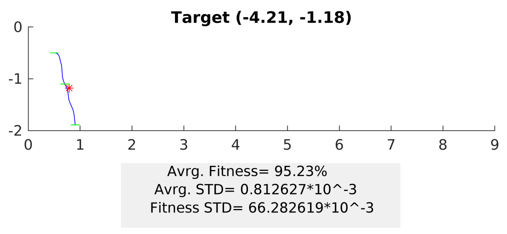
Target point 2



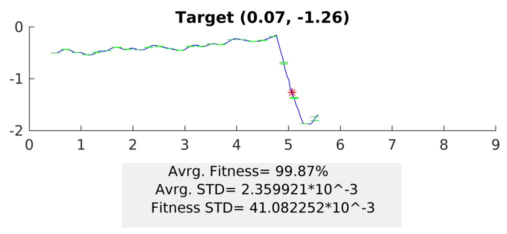
Target point 3



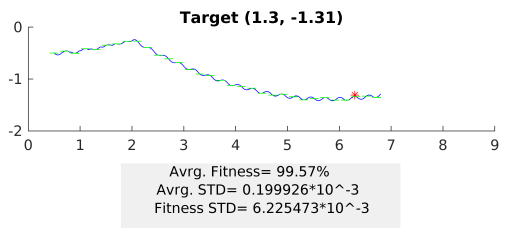
Target point 4



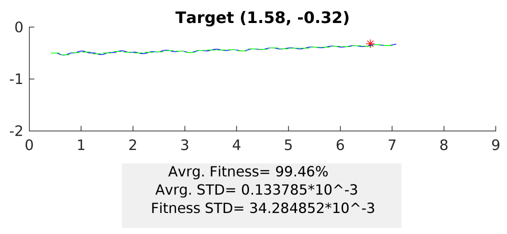
Target point 5



Target point 6



Target point 7



Target point 8

Figure 7. Trajectories (mean and standard deviation at each time step, across 10 repetitions) of the best evolved agent for 8 (out of 10) target points.