An In-Memory Physics Environment as a World Model for Robot Motion Planning

Navin Ipe[1]

[1]M.S.Ramaiah University of Applied Sciences

March 29, 2024

Abstract

Publication reference: https://doi.org/10.1007/978-981-16-1740-9_46

This paper investigates the utilization of a physics simulation environment as the imagination of a robot, where it creates a replica of the detected terrain in a physics simulation environment in its memory, and “imagines” a simulated version of itself in that memory, performing actions and navigation on the terrain. The physics of the environment simulates the movement of robot parts and its interaction with the objects in the environment and the terrain, thus avoiding the need for explicitly programming many calculations.

An In-Memory Physics Environment as a World Model for Robot Motion Planning

Navin K Ipe

***Abstract*—Conventional robots are capable of detecting terrain and creating a 2D or 3D map of the terrain in memory, which is utilized by the robot's algorithms to plan navigation. Such algorithms are primarily focused on path optimality, gaits and joint positioning. This paper investigates the possibility of utilizing a physics simulation environment as the imagination of a robot, where it creates a replica of the detected terrain in a physics simulation environment in its memory, and "imagines" a simulated version of itself in that memory, performing actions and navigation on the terrain. The physics of the environment simulates the movement of robot parts and its interaction with the objects in the environment and the terrain, thus avoiding the need for explicitly programming many calculations. The robot chooses the best possible action from multiple simulations of movement, and executes it in the real world. Moreover, as the complexity of motion increases with each degree of freedom of the robot's joints, this paper also explores the utility of uniform pseudo-randomness to explore the fitness landscape of robot motility, and compares it with Computational Intelligence algorithms. Such techniques could potentially simplify the algorithmic complexity of programming multi-jointed robots, and also be capable of dynamically adjusting the "mental" simulation of the robot when it encounters environments with different gravity, viscosity or traction, merely by adjusting parameters of the simulated environment.**

***Index Terms*—Machine Learning, Computational Intelligence, Differential Evolution, Particle Swarm Optimization, Robotics, Uniform Randomness.**

I. Introduction

THE animal brain is capable of constructing a complex world-model of the various sensory inputs it perceives. Within this imagined world, it is capable of simulating various actions by utilizing memories and even mixing various senses and past experiences. Imagination, thus offers a powerful method of evaluating various actions and scenarios before executing them.

In the realm of machine learning, mathematical modeling of phenomena has been a preferred method of forming approximations that explain and predict phenomena. The decimal number system, complex numbers, statistics, differential equations, probability, Eigen and Fourier transforms are some such mathematical methods. Biological brains appear to model phenomena a bit differently, using specialized cells and vast data storage [1], [2] to record and predict phenomena via an imagination that accounts for context and estimates expected outcomes at frequent intervals of time. Learnings from prior investigations into intelligence [3] led to the conclusion that the creation of an intelligent machine necessitated the machine to be an embodied consciousness that could experience the world around it. Only then, would the machine be capable of associating its experiences with the experiences of humans, thus helping it understand the meaning of objects, phenomena and words. As a first step toward building such capabilities, this paper investigated the possibility of simulating some aspects of the real world via a 2D physics simulation environment. A robot with a body composed of a rectangular chassis and multiple limbs connected to rotary joints and motors, is assigned the task of navigating through various obstacles in the environment, until it reaches its goal of crossing a yellow finish-line. The unique aspect of this exercise is the manner in which the robot creates a replica of the "real-world", in its memory as an "imaginary world", and is allowed to imagine itself performing various motor actions to move its limbs in various ways in the imaginary world, to check which of those motions lead to the best possible motion that could take it forward. When the best motion was selected, the robot performed that motion in the "real-world". An interesting phenomenon noted, was that when the robot body and limbs were in a certain start position, each time the motors were assigned the same set of motor rates that they were assigned in the previous attempt, there was no guarantee that the robot would move in the same way as it did before. This was a hurdle because it lacked repeatability and reliability, but it also offered a good approximation of the various forces and dependencies that act on objects in the biological world, hence roughly modelling unpredictable motion. The reason for this unpredictability was also investigated, because any research work tackling robot movements in the real or simulated world, has to account for this phenomenon and be resilient to unpredictable terrain.

Section II presents similar work that require a legged robot to reach a goal, Sect. III presents the design decisions and test environment, Sect. IV investigates why the robot does not consistently perform the expected motion, Sects. V and VI present the reasoning behind the trial runs and the inferences derived. The paper concludes with Sect. VII and offers some tips for future work.

II. Related Work

Various attempts have been made at building embodied and context-aware architectures [4], [5]. Artificial Neural Networks (ANN) are available in a large variety of structures to store memory with respect to various contexts [6] and various attempts have been made to utilize Computational Intelligence

Author acknowledges the support received from M S Ramaiah University of Applied Sciences, Bengaluru, India. N. K. Ipe is with the Department of Computer Science and Engineering, M S Ramaiah University of Applied Sciences, Bengaluru, India (e-mail: navinipe@gmail.com).

Manuscript received: Aug 18, 2020

(CI) in robot gaits [7]. However, these techniques are yet to achieve success in solving the large problem space posed by real-world situations. This is especially true of ANN's which depend so heavily on trained weights, that over-fitting, under-fitting and the vanishing gradient problem [8] pose serious limitations. Even concepts like neural network dropouts or Long Short Term Memory (LSTM) afford the algorithm only a limited memory, fixed inputs and outputs which are insufficient to represent complex phenomena, some of which need more prominent representation than others, as evidenced by human homunculi [9].

In order to model space, objects and generate an imagination without utilizing complex mathematical approximations, the animal brain may offer clues, where spatial perception is modeled similar to a Geographical Information System (GIS) software. The brain has specialized sets of cells to handle proprioception [10] and the vestibular system. To understand and process spatial information, place cells play a role [11], dead reckoning [12], spatial view cells [13], grid cells [14], border cells [15], speed cells [16] and head direction cells [17] too. Oddly, despite this vast knowledge, machine learning (ML) algorithms are yet to model these concepts accurately. Various researchers have attempted modeling imagination in the memories of robots [18], [19], but the robots and the environments chosen were simple, and CI does not appear to be used for imagination. Techniques that utilize CI algorithms tend to focus on navigation, path planning, locomotion [20]–[22] and joint positioning [23]. CI algorithms have also been used for navigation on uneven and varied terrain [24]–[26], while Gaussian and probabilistic models have been used to decide appropriate footholds on terrain [27]. ANN's have also been used to provide controller action via a multi-objective differential evolution (DE) method [28].

III. Design Decisions

1) Robot and environment:

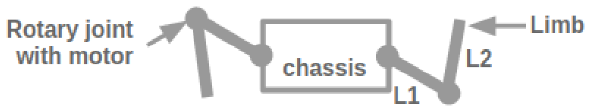


Fig. 1: Robot

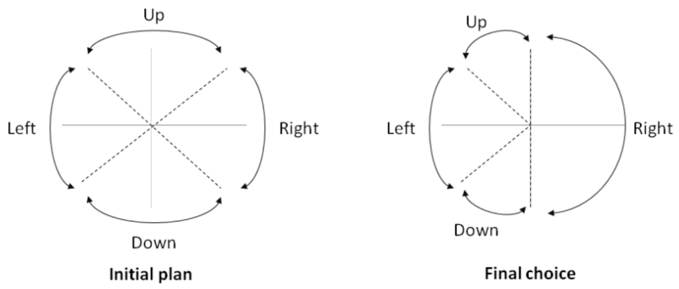


Fig. 2: Direction angles for movement

In order that a robot be versatile in tackling random, uneven terrain, the limbs of the robot need to move at any desired angle. A limb coupled with a second limb, can even produce motion similar to a linear piston motion. In order to prevent unnecessary tumbles on uneven terrain, a rectangular chassis was chosen instead of a circular one. The general design of the robot is shown in Fig. 1. Two motors are attached to the chassis, and limbs $L1$ are attached to each motor. A second set of motors are attached to the free ends of the limb, to which limbs $L2$ are attached. The rotation of the motor attached to the chassis causes $L1$ to move and inadvertently causes $L2$ to move along with it. Motor rates range from 0 to ± 6 (no unit). More limbs can be added to the robot if necessary. The robot was designed to tumble, rather than "walk".

The robot was initialized in a 2D physics simulation environment named PyMunk, where the robot's navigation capabilities can be tested on flat ground as well as on obstacles of various shapes and sizes. The environment was initialized with an acceleration due to gravity of $9 \mathrm{m/s}$, the robot chassis, with a weight of $5kg$ and the robot limbs with weights of $0.5kg$ each. There was no option to assign a weight to the motors, so they are assumed to be weightless. Directly above the environment (called the "real world"), another duplicate environment is initialized. This second environment is called "imaginary world", and represents how the robot perceives the real world. The robot "imagines" its actions in the imaginary world by simulating motion of multiple replica's of itself. Ideally, the terrain in the imaginary world should have been created based on the sensory perception of the robot, but for simplicity, all objects in the real world were directly duplicated in the imaginary world. The imaginary world was meant to simulate the manner in which biological creatures can imagine performing various scenarios in their mind, before choosing what they think is the best possible action and then executing it.

Ideally, the robot should perform actions and assess them at fractions of a second, due to the dynamic nature of the real world and the interactions of various body parts of the robot. However for simplicity and to avoid computational load, it was decided to perform actions for one full second before performing any assessments. The robot was given four possible directions of motion: up, down, left and right, based on angles of equal proportion. However, during initial tests, it was realized that right-ward motions (which would take the robot closer to the finish line) also consisted of up and down motions when navigating terrain, so a broader angle was assigned to the rightward motion as shown in Fig. 2. The robot was programmed only to deal with static environments, therefore it was possible to execute robot motions for a full second. An environment with moving objects would have required frequent checks and feedback control loops, which was not within the scope of this paper, so such mechanisms were avoided. During initial tests, a reverse-and-repeat behaviour was also programmed into the robot, wherein it would perform an action, and if the action did not move it forward, the motor rates would be reversed to bring the robot back in position. This behaviour was switched off eventually, to be able to perform a fair comparison between uniform random, DE and

PSO algorithms.

Five worlds with varied terrains were initially created for testing. A world with a flat ground, a world with randomly distributed rectangular obstacles of varying sizes (also tests the possibility of the robot getting stuck within concave blockades, notches in terrain and pits), a world with randomly distributed spheres of varying sizes (tests the robot's capability to handle curved surfaces), a world with a large staircase and a world with alternating gaps that require the robot to climb a high obstacle and then squeeze through a tunnel-like gap. The staircase terrain was eventually discarded as it didn't add value compared to the rectangular obstacles. The terrains are depicted in Fig. 3. All surfaces in the PyMunk environment were assigned a unitless friction of value 20. The algorithm could be run for n number of trials, and for each trial of each terrain that required random obstacles, the obstacles were randomly generated and saved on disk. So when the same terrain needed to be run again with another algorithm, it could be loaded from disk and used. This helped ensure that various algorithms were tested under the same terrain, while also ensuring that each trial had a different terrain. All objects within the terrain were fixed in position and unmovable.

A few attempts were also made to test the capability of the robot using a single leg with two limbs and two legs with a single limb. Both of these performed sub-optimally. Utilizing three limbs per leg offered much better dexterity and the capability to grip objects, but for simplicity, two legs with two limbs each were finalized for experimentation.

A. *Storage and Algorithms*

An initial attempt was made to give the robot a memory, by using a graph datastructure where each node would be composed of a hash of the chassis angle and all the limb angles. The hash would allow locating the right node instantly, given any robot position. The edges of the graph would be the distance the robot moved and the motor rates used. Such a graph was capable of storing memories of previous movements and positions, but had limitations of memory, since the number of possible angles for the chassis and limbs themselves were $360^5 = 6.04 \times 10^{12}$ nodes, where 360 is the number of degrees of the possible angles and 5 represents the chassis and the four limbs. The multiple edges between nodes of the graph was even higher. As the number of nodes increased, it weighed down heavily on memory. Additionally, since the nodes were dependent on robot angles, each different type of terrain would result in a different type of motion, so it was impossible to predict motion using merely the information in the graph, unless even tactile information was also incorporated into the identity of the nodes. This would cause an exponential increase in the number of nodes required for memory. The other limitation such a graph node had, was the inconsistency caused by the physics environment, as explained in IV. These limitations led to the decision of not storing memory in a graph, but instead, performing new calculations for each movement. In hind-sight, this is similar to how biological creatures tackle terrain too. At each fraction of a second, new calculations are performed in the creature's imagination about the outcome of any new action to perform, based on the perceived world.

Algorithm 1 Robot motion

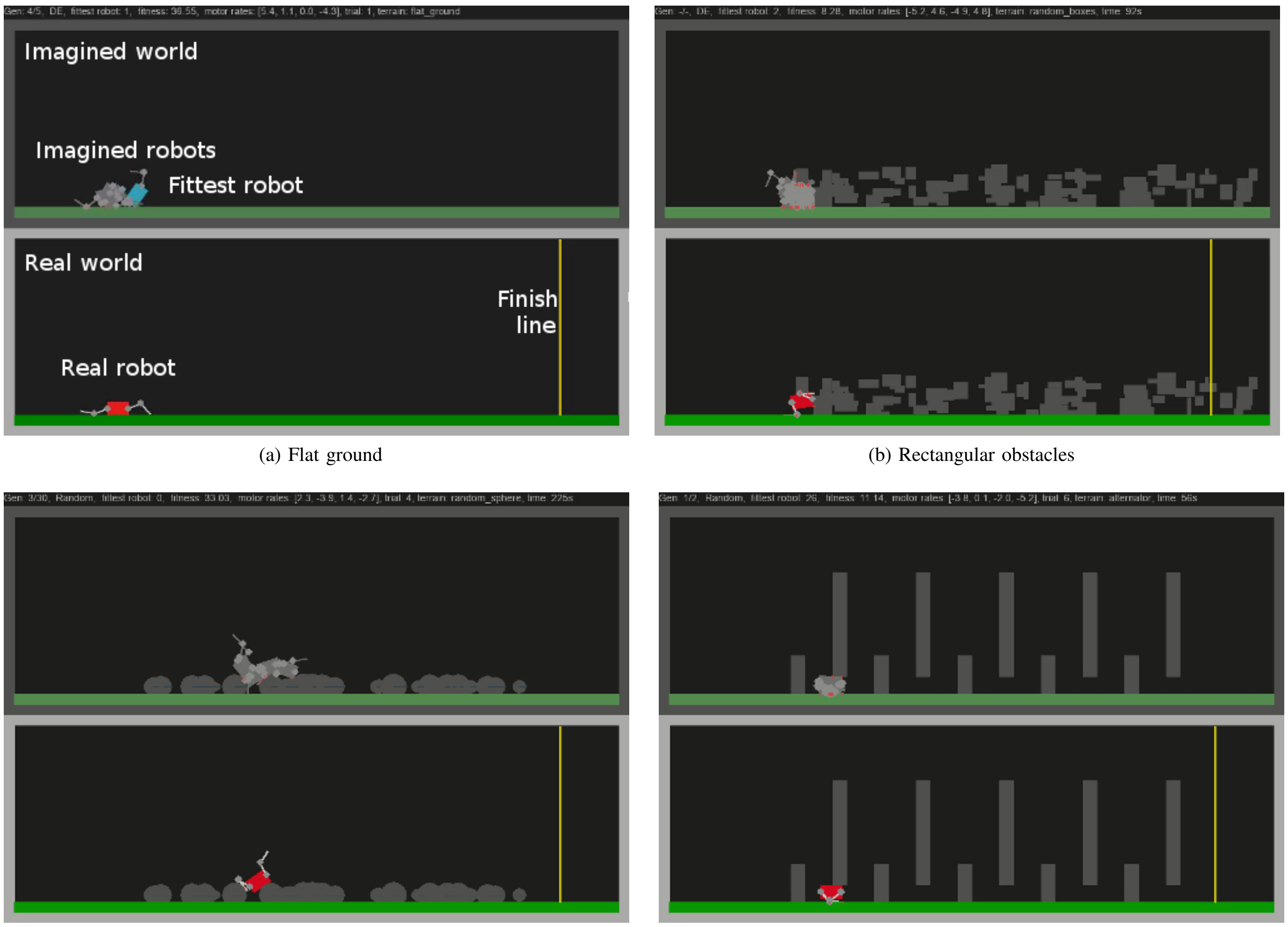
```
Step 1: Initialize $R$. Initialize $s = 0$.
Step 2: Initialize $I$ with angles of $I$ = angles of $R$.
Step 3: Store $(x, y)$ positions of $R$ as $P_R$.
Step 4: $For\ gen = 1\ to\ g$
Step 5: Assign motor rates to $I$ based on random values or Algorithm 2 or Algorithm 3 and run for one second.
Step 6: Store motor values of $I_F$ and set angles of $I$ = angles of $R$.
Step 7: $Next\ gen$
Step 8: Set angles of $R$ = angles of $I_F$ and run for one second.
Step 9: If $P_R$ variation $< 20$ pixels, $s = s + 1$, else $s = 0$.
Step 10: If $s == 5$ goto Step 11 else goto Step 12.
Step 11: Set random angles for $R$. Move motors for one second. Repeat Step 11 five times.
Step 12: If $x$ of $P_R > L_x$, end program else goto Step 2.
```

The simplest way to describe the algorithm is as such: Robots are initialized in the imaginary world with the same angles and position as the real robot R. Each imaginary robot $I = \{I_1, I_2 \ldots, I_p\}$ of population size p is allowed to move its limbs in various directions and motor rates for one second. Such attempts are made for g generations. The motor rates generating the greatest magnitude of movement in the positive x direction (the fittest robot I_F) are utilized to move the real robot, and the real robot's limbs are moved for one second. If the robot is stuck in the same position for five consecutive attempts, the algorithm switches to a state where it performs a random motion for each second, for five seconds. This was found to be very effective to get the robot un-stuck. Once the robot is un-stuck, the algorithm switches to the normal imagination motion followed by real robot motion, until the robot crosses the finish line L_x. All random values used were generated from uniform random functions based on the Mersenne Twister pseudo-random number generator.

The differential evolution (DE) algorithm presented as Algorithm 2 and the particle swarm optimization (PSO) algorithm presented as Algorithm 3 are a little different from convention, since the fitness values could be calculated only after all I robots had completed motion. The algorithms were also given an $rr = 0.1$ probability (determined by a uniform random function $random(0, 1)$ ranging from values 0 to1) to randomly initialize motor rates of 10% of the robots. In PSO, the fitness of a robot at time t is $I_f(t)$, and it is compared with the fitness values calculated in the previous run of PSO, depicted as $I_f(t-1)$. Also stored, are the personal best motor rates, depicted by b_p.

IV. Physics inconsistency

Objects in simulated 3D and 2D physics environments tend to have slightly unpredictable or jittery motion, sometimes causing objects to move with an explosive velocity. A similar slight inconsistency was noticed when the fittest robot I_F sometimes moved a little differently for the exact same motor



```
Gen: 4/5,  DE,  fittest robot: 1,  fitness: 38.55,  motor rates: [5.4, 1.1, 0.0, -4.3], trial: 1, terrain: flat_ground
Gen: -/-,  DE,  fittest robot: 2,  fitness: 8.28,  motor rates: [-5.2, 4.6, -4.9, 4.8], terrain: random_boxes, time: 92s
Gen: 3/30,  Random,  fittest robot: 0,  fitness: 33.03,  motor rates: [2.3, -3.9, 1.4, -2.7], trial: 4, terrain: random_sphere, time: 225s
Gen: 1/2,  Random,  fittest robot: 26,  fitness: 11.14,  motor rates: [-3.8, 0.1, -2.0, -5.2], trial: 6, terrain: alternator, time: 56s
```

(a) Flat ground

(b) Rectangular obstacles

(c) Spherical obstacles

(d) Alternating obstacles

Fig. 3: The 2D simulation environments

Algorithm 2 Modified Differential Evolution

```
Step 1: Determine $I_F$.
Step 2: Set $rr = 0.1$, $cr = 0.3$, $\beta_v = 2$, $r_\beta = 1/40$.
Step 3: $For\ each\ I_p\ in\ I$
Step 4: If $I_p == I_F$, goto Step 3 else goto Step 5.
Step 5: Randomly select 3 robots $I_1, I_2, I_3$ from $I$, excluding $I_F$.
Step 6: Get motor rates of selected robots $r_p, r_1, r_2, r_3$.
Step 7: If $random(0,1) > rr$, set random motor rates and goto Step 10 else goto Step 8.
Step 8: If $random(0,1) \leq cr$, motor rates = $r_1 + round\left(\beta_v \times (r_2 - r_3)\right)$.
Step 9: If motor rates out of range of $\pm 6$, motor rates = $r_p$, else $r_p$ =motor rates.
Step 10: If $\beta_v > r_\beta$, $\beta_v = \beta_v - r_\beta$.
Step 11: $Next\ I_p$
```

Algorithm 3 Modified Particle Swarm Optimization

```
Step 1: Determine fittest robot $I_F$ and fitness $I_f$ for all $I$.
Step 2: Set $rr = 0.1$, $c_1 = 1$, $c_2 = 2$. Set velocities $v$ and personal best rates $b$ for all robots to 0.
Step 3: $For\ each\ I_p\ in\ I$
Step 4: if $I_p == I_F$, goto Step 3 else goto Step 5.
Step 5: Get motor rates of current robot $r_p$ and fittest robot $r_F$.
Step 6: If $random(0,1) > rr$, set random motor rates and goto Step 13 else goto Step 7.
Step 7: If $I_f\,(t) > I_f(t-1)$, $I_f(t-1) = I_f\,(t)$ and $b_p = r_p$.
Step 8: $C = c_1 \times random(0,1) \times (b_p - r_p)$.
Step 9: $S = c_2 \times random(0,1) \times (r_F - r_p)$.
Step 10: $v_p = v_p + C + S$.
Step 11: $r_p = r_p + v_p$
Step 12: Clamp values of $v_p$ and $r_p$.
Step 13: $Next\ I_p$.
```

rate and position. In order to investigate the cause of this phenomenon, a separate test environment was setup to measure the variation in the start and end positions of the robot after one second of motion according to a specific motor rate (the robot moves for fifty frames each second). The robot was re-initialized to the same start position at each repetition and run with the same motor rate. This was repeated a hundred times.

Twenty such trials were performed, with each trial having different motor rates. The dependence of the robot position was checked using the motor rates, the chassis angles, the limb angles and the contact made by each limb with the ground. Fig. 4 depicts correlations performed, where Fig. 4a shows a box-plot of dx (pixel distance from start position to end position in x axis) and dy (pixel distance from start position to end position in y axis) for 100 simulations of each trial. Figure 4b allows checking if the wide variations in dx had anything to do with the motor rates being high. Sufficient correlation with motor rates was not found. The angles and contact with the ground (Limb touch) were examined from multiple trials, where, as a sample, Figs. 4c and 4d depict a consolidation of a hundred trials each, shown by dots of a particular colour where the x axis represents the number of frames in each second. In each of the Touch figures in figures like Fig. 4c, a value of 0 means no contact was made with the ground, and any value of 1 or more means that the corresponding limb made contact with the ground at those many points on the limb, per frame (multiple parts of the limb can be touching the ground at certain points of time).

Since the inconsistencies in movement could not be correlated to the motor rates or the angles or the contact made with the ground, a correspondence was initiated with the creator of PyMunk, during which a series of experiments performed, highlighted two issues.

1) Internal caches retain some state between steps, causing different motion even when the same motor rate is executed. This could be solved by deleting the robot and re-creating it.
2) Even when robot is deleted and re-created, the torque exerted when it pushes onto an object, can sometimes cause explosive motions, where the robot parts briefly get separated from their joints. This could possibly be due to a coordinate, force, angle or number becoming NaN, Inf or a very large number. This issue could potentially be mitigated by setting a max force on the motors and checking the impulse on the joints to detect when it is too high.

Although only the possible causes of the unpredictability could be hypothesized, these findings imply that conventional machine learning techniques that depend on consistent outputs to "learn", would benefit from accounting for the uncertainties in such simulation environments. A plus point being that since the biological world is highly multi-dimensional and unpredictable, the uncertainties in the simulation environment could help in designing resilient algorithms.

V. Trials

As mentioned in Sect. III-A, the robots were initially run with a learning component, where various motions were stored in a graph, but since the graph required tactile information and a very large storage, the approach was deemed impractical and the robots were run by storing only each robot's motor rates and positions as memory. Trials were conducted on the various available terrains, where the objective of the robot was to maximize the distance it moved in the positive x direction in each attempt, to reach a finish line at the end of the environment. Trial V-C was programmed with the added functionality of moving randomly to get out of a stuck position. The time taken by each robot is shown in Table I.

A. *Trials with* $g = 4$, $p = 5$

On flat ground, a small population and few generations showed good results, and the uniform random function was able to consistently locate better global optima, compared to DE and PSO. However, in the rectangular obstacles terrain, locating optimal solutions was more difficult, and the robot took longer to reach the finish line. As a result, a decision was taken to expand the number of generations to 30.

B. *Trials with* $g = 30$, $p = 5$

Not surprisingly, running a greater number of generations helped locate better global optima and helped the robot reach the finish line faster in the flat terrain, the rectangles terrain and the spheres terrain. During this trial run, the results of the spheres terrain brought about an assumption that running 30 generations may not really be required, and fewer generations would suffice if a larger population could explore larger expanses of the fitness landscape. Therefore, the alternator terrain trial was not run, and instead trials with two generations and a population size of 30 were initiated.

C. *Trials with* $g = 2$, $p = 30$

One of the reasons the rectangles terrain took longer than usual, was the fact that robots tended to get stuck in concave spaces at the beginning of the terrain, before getting unstuck and moving toward the finish line. One of the reasons they got stuck, was the sub-optimal identification of the best motion, due to fewer generations. Trial V-B had $30 \times 5 = 150$ searches on the fitness landscape per attempt, but the current trial had only $2 \times 30 = 60$ searches. It was also interesting to note a trend that in all trials until this stage, the uniform random algorithm gave a better result than DE or PSO in a majority of the runs.

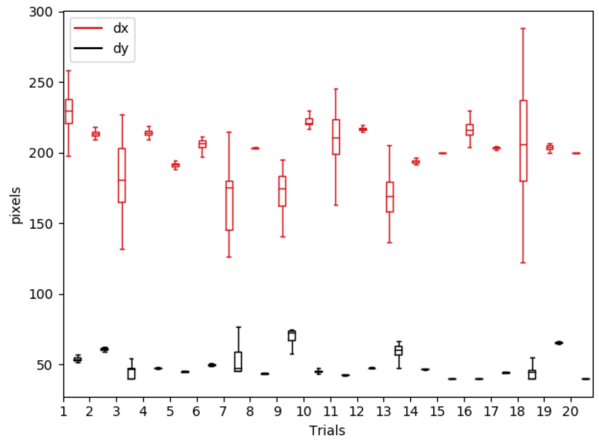
D. *Trials with* $g = 30$, $p = 30$

In order to improve the search for a better global optimum, the number of generations and population were kept at 30. A higher number would have been more beneficial, but was limited at 30, keeping in mind the computational load. The searches per attempt reached $30 \times 30 = 900$, which helped locate better movements at each attempt (albeit plagued by the physics inconsistencies mentioned in Sect. IV).

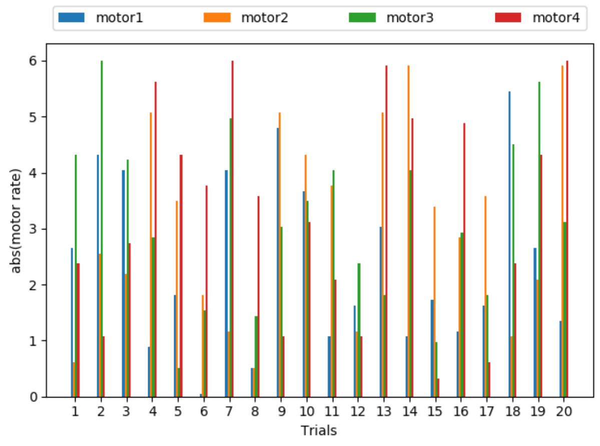
VI. Inferences

A. *Uniform randomness is effective*

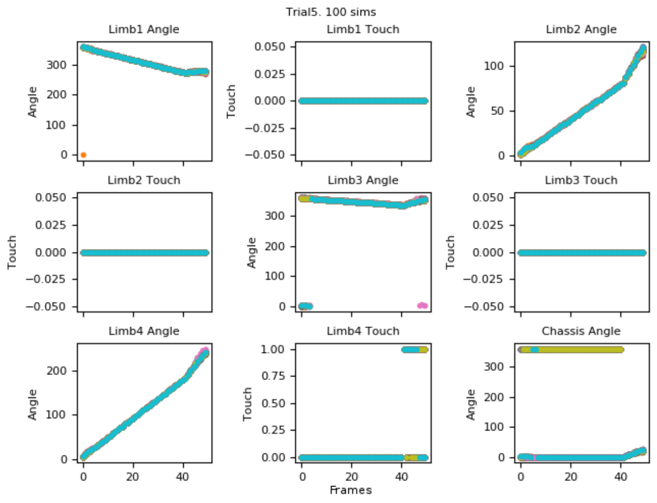
The objective of using CI algorithms, is to locate a near-optimal solution in a fitness landscape by exploiting areas with good fitness. However, when the fitness landscape is



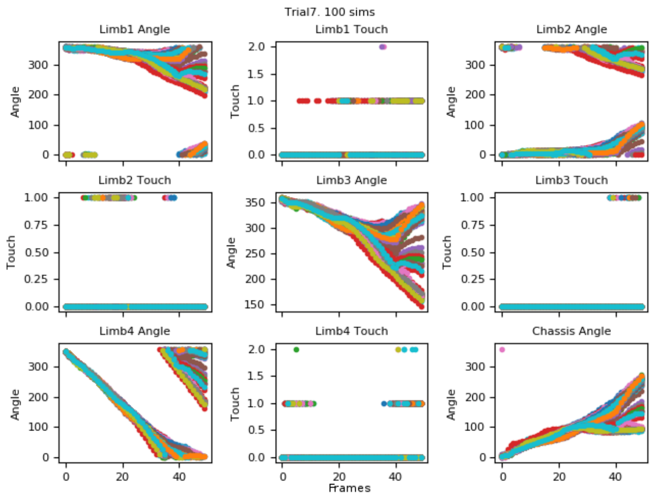
(a) dx, dy variations



(b) Motor rates



(c) Angles and contact points in each frame of trial 5



(d) Angles and contact points in trial 7

Fig. 4: Determining the cause of variation in motion patterns

extremely huge and computational resources are scarce, it may be simpler and more beneficial to utilize uniformly random functions. Table I lists the average time taken for the real robot to reach the finish line. Utilizing the values that were obtained before averaging, a hypothesis test was designed to test whether there was a significant difference or advantage to using CI algorithms, as compared to uniform pseudo-randomness. The null hypothesis H_0 was that in highly multi-dimensional fitness landscapes, uniform pseudo-randomness could locate equally good local optima as the CI algorithms, thus resulting in completion times that are more-or-less similar. The alternate hypothesis H_1 was that CI algorithms would produce a significant improvement in results since they explore the fitness landscape near any local optima, while continuing to explore globally too, so the completion times of DE and PSO should show a significant difference from the uniformly random results.

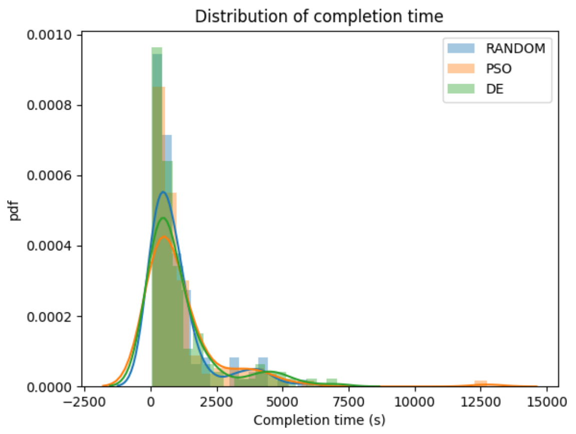


Fig. 5: Probability density functions of Random, DE and PSO

Figure 5 depicts the positive skewed distribution of the time the real robot took to reach the finish line. The skews were 2.13, 2.16 and 3.77 for random, DE and PSO respectively. A 0.05 confidence interval was considered when utilizing a one-

TABLE I: Trials

gen. (g)	Popu. (p)	Terrain	Algo.	Avg. time (s) for 10 trials
5	4	Flat	Random	18.43
			DE	31.38
			PSO	24.8
		Rectangles	Random	143.12
30	5	Flat	Random	13.6
			DE	16.77
			PSO	17.29
		Rectangles	Random	46.34
			DE	51.01
			PSO	62.79
		Spheres	Random	28.99
			DE	28.56
			PSO	27.42
2	30	Flat	Random	22.17
			DE	24.33
			PSO	28.4
		Rectangles	Random	115
			DE	90.43
			PSO	144.13
		Spheres	Random	43.0
			DE	38.17
			PSO	45.73
		Alternator	Random	210.2
			DE	248.23
			PSO	214.9
30	30	Flat	Random	18.08
			DE	17.05
			PSO	17.51
		Rectangles	Random	67.36
			DE	82.56
			PSO	101.47
		Spheres	Random	36.0
			DE	41.85
			PSO	46.95
		Alternator	Random	132.55
			DE	146.5
			PSO	130.71

way Mann-Whitney rank to test the hypothesis, resulting in $p-value = 0.378$ for random versus DE and $p - value = 0.241$ for random versus PSO. H_0 was not rejected, thus proving that the utilization of uniformly random numbers instead of CI algorithms can be considered a viable option. A larger number of generations and a larger population of imagined robots can help locate better global optima, resulting in more efficient and effective robot locomotion.

B. Physics simulations are a viable alternative

The fitness landscape for a robot that can move its limbs at any angle, is highly multi-dimensional. Each limb could move with one among 130 motor rates, ranging between values of -6 and $+6$ (no unit). Each $L1$ limb's motion caused $L2$'s motion and from each such inadvertent $L2$ position, $L2$ could perform its own motion. Besides, each motor motion begins at one of 360 possible limb angles, and the motion of the robot depends on which limb makes contact with the terrain. These are affected by the number of contact points, motor rates and angles of contact. Given such a vast range of motion and possibilities, it is evident why biological creatures have a limited range of motion, and why certain specific motions are repeated frequently, even though they may not be optimal. Robots would also benefit from storing favorite motions, and possessing the capability to move the limb back to a starting position for a chosen motion, rather than continuing motion from an existing position. The fact that the robot successfully reached the finish line by selecting the motor rates it imagined in a simulated replica of the real world, shows that such simulations could indeed be considered a viable alternative to explicit programming, since when the fitness landscape is so vast for such a simple robot, the complexity of programming the movements for highly multi-jointed robots would be a much more cumbersome task, especially given the various terrains and varied effects of gravity, viscosity or buoyancy it would encounter. Normally, physics simulations incorporate delays, to allow simulated objects to interact with each other in realtime. So a robot would move a certain distance in one second. As an improvement to current methods, the physics simulation algorithms could be re-designed to calculate all physical interactions of that one second, in a fraction of a second, thus allowing multiple simulations of a large population of robots to be performed in fractions of a second, multiple times.

VII. Conclusion

This paper demonstrated that an embodied consciousness (a robot) could navigate various kinds of terrains and surfaces, by imagining various possibilities of the action and terrain in memory, by simulating the physics of the actions. Moreover, it is not necessary to utilize CI algorithms to locate global optima in a highly multi-dimensional fitness landscape (in-fact, on many occasions, DE and PSO's converging nature led to the robot getting stuck at local optima). A simple uniform random number generator is capable of locating the global optimum.

It is also important to note the following, for future work:

1) Checks and balances are necessary every fraction of a second, to ensure that objects in a physics simulation do not react adversely during interaction with each other.
2) Natural phenomena are simple, but appear complex due to being highly interconnected. Rather than create complex models to represent this complexity, it helps to observe anomalies and utilize simple concepts to grow complex solutions.
3) Biological processes are optimized to conserve energy. This builds into the system, a fatigue or laziness, and a need to settle for a local optimum (sub-optimal solution to any given problem). Robots do not necessarily require a focus on optimization and efficiency. Robots can be designed to utilize abundantly available resources, allowing the robot to explore and experiment freely. The molecular levels of Nature seem to be designed as such – with an abundance that simultaneously allows wastefulness and efficiency, thus allowing discovery of new possibilities in a vastly multi-dimensional universe.
4) Higher forms of biological intelligence are intelligent, not just due to high data storage capability, but also due to performing multiple iterations of associations and correlations between stored data and then performing trial and error experiments to vary the outcome of observations. This is done over a period of millions of years.

These are a crucial task that simplistic algorithms like ANN cannot perform effectively, due to being dependent on neural weights, rather than depending on a database containing a lifetime of acquired information. A full-fledged event-based memory for intelligent machines however [29], could augment or even replace the need for utilizing physics simulations to create an in-memory world-model.

References

[1] F. Sargolini, M. Fyhn, T. Hafting, B. L. McNaughton, M. P. Witter, M.-B. Moser, and E. I. Moser, "Conjunctive representation of position, direction, and velocity in entorhinal cortex," *Science*, vol. 312, no. 5774, pp. 758–762, 2006. I

[2] D. Nikolić, "The brain is a context machine," *Review of psychology*, vol. 17, no. 1, pp. 33–38, 2010. I

[3] N. Ipe, "Facts and anomalies to keep in perspective when designing an artificial intelligence," 2020. I

[4] I. Kotseruba, O. J. A. Gonzalez, and J. K. Tsotsos, "A review of 40 years of cognitive architecture research: Focus on perception, attention, learning and applications," *arXiv preprint arXiv:1610.08602*, pp. 1–74, 2016. II

[5] P. Ye, T. Wang, and F.-Y. Wang, "A survey of cognitive architectures in the past 20 years," *IEEE transactions on cybernetics*, vol. 48, no. 12, pp. 3280–3290, 2018. II

[6] A. Tchircoff, "The mostly complete chart of neural networks, explained," *Towards Data Science*, pp. 1–29, 2017. II

[7] C. Rong, Q. Wang, Y. Huang, G. Xie, and L. Wang, "Autonomous evolution of high-speed quadruped gaits using particle swarm optimization," in *Robot Soccer World Cup*. Springer, 2008, pp. 259–270. II

[8] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998. II

[9] W. Penfield and E. Boldrey, "Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation," *Brain*, vol. 60, no. 4, pp. 389–443, 1937. II

[10] J. Winter, T. J. Allen, and U. Proske, "Muscle spindle signals combine with the sense of effort to indicate limb position," *The Journal of physiology*, vol. 568, no. 3, pp. 1035–1046, 2005. II

[11] J. O'Keefe, N. Burgess, J. G. Donnett, K. J. Jeffery, and E. A. Maguire, "Place cells, navigational accuracy, and the human hippocampus," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 353, no. 1373, pp. 1333–1340, 1998. II

[12] I. Q. Whishaw, D. J. Hines, and D. G. Wallace, "Dead reckoning (path integration) requires the hippocampal formation: evidence from spontaneous exploration and spatial learning tasks in light (allothetic) and dark (idiothetic) tests," *Behavioural brain research*, vol. 127, no. 1-2, pp. 49–69, 2001. II

[13] E. T. Rolls, "Spatial view cells and the representation of place in the primate hippocampus," *Hippocampus*, vol. 9, no. 4, pp. 467–480, 1999. II

[14] C. F. Doeller, C. Barry, and N. Burgess, "Evidence for grid cells in a human memory network," *Nature*, vol. 463, no. 7281, pp. 657–661, 2010. II

[15] C. Barry, C. Lever, R. Hayman, T. Hartley, S. Burton, J. O'Keefe, K. Jeffery, and N. Burgess, "The boundary vector cell model of place cell firing and spatial memory," *Reviews in the Neurosciences*, vol. 17, no. 1-2, p. 71, 2006. II

[16] E. Kropff, J. E. Carmichael, M.-B. Moser, and E. I. Moser, "Speed cells in the medial entorhinal cortex," *Nature*, vol. 523, no. 7561, pp. 419–424, 2015. II

[17] J. S. Taube, R. U. Muller, and J. B. Ranck, "Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis," *Journal of Neuroscience*, vol. 10, no. 2, pp. 420–435, 1990. II

[18] C. Blum, A. F. Winfield, and V. V. Hafner, "Simulation-based internal models for safer robots," *Frontiers in Robotics and AI*, vol. 4, p. 74, 2018. II

[19] S. Rockel, D. Klimentjew, L. Zhang, and J. Zhang, "An hyperreality imagination based reasoning and evaluation system (hires)," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5705–5711. II

[20] B. Tang, Z. Zhu, and J. Luo, "Hybridizing particle swarm optimization and differential evolution for the mobile robot global path planning," *International Journal of Advanced Robotic Systems*, vol. 13, no. 3, p. 86, 2016. II

[21] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019. II

[22] Z.-Y. Yang and C.-F. Juang, "Evolutionary locomotion control of a hexapod robot using particle swarm optimized fuzzy controller," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2014, pp. 3861–3866. II

[23] N. Rokbani, E. Benbousaada, B. Ammar, and A. M. Alimi, "Biped robot control using particle swarm optimization," in *2010 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2010, pp. 506–512. II

[24] O. Janrathitikarn and L. N. Long, "Gait control of a six-legged robot on unlevel terrain using a cognitive architecture," in *2008 IEEE Aerospace Conference*. IEEE, 2008, pp. 1–9. II

[25] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008. II

[26] Q. Nguyen, A. Agrawal, X. Da, W. C. Martin, H. Geyer, J. W. Grizzle, and K. Sreenath, "Dynamic walking on randomly-varying discrete terrain with one-step preview." in *Robotics: Science and Systems*, vol. 2, no. 3, 2017. II

[27] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard, "Learning predictive terrain models for legged robot locomotion," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3545–3552. II

[28] J. Teo and H. A. Abbass, "Coordination and synchronization of locomotion in a virtual robot," in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02.*, vol. 4. IEEE, 2002, pp. 1931–1935. II

[29] N. Ipe, "Context and event-based cognitive memory constructs for embodied intelligence machines," 2020. 4