OPTIMIZING A QUADRUPED ROBOT: A COMPARISON OF TWO METHODS

ROBERT PASTOR[1], ZDENKO BOBOVSKY[1], PETR OSCADAL[1], JAKUB MESICEK[2], MAREK PAGAC[2], ERIK PRADA[3], LUBICA MIKOVA[3], JAN BABJAK[1]

[1]Department of Robotics, [2]Department of Machining, Assembly and Engineering Metrology, Faculty of Mechanical Engineering, VSB-TU Ostrava, Czech Republic.
[3]Department of Mechatronics, Faculty of Mechanical Engineering, Technical University of Kosice, Slovak Republic

DOI : 10.17973/MMSJ.2021_6_2021008

robert.pastor@vsb.cz

Robots that have been optimized in simulation often underperform in the real world in comparison to their simulated counterparts. This difference in performance is often called a reality-gap. In this paper, we use two methods, genetic algorithm and topology optimization, to optimize a quadruped robot. We look at the original and optimized robots' performance in simulation and reality and compare the results. Both methods show improvement in the robot's efficiency, however the topology optimization behaves in a more predictable manner and shows similar results in simulation and in real laboratory testing. Modifying robot morphology with a genetic algorithm, although less predictable, has a potential for more improvement in efficiency.

KEYWORDS

reality-gap, genetic, topology, optimization, quadruped

1 INTRODUCTION

When designing a robot, its morphology, sensory apparatus, motor system, control architecture, and other aspects need to be considered. All of these aspects interact and determine robot's behavior [Doncieux 2015]. Evolutionary robotics (ER) is a holistic approach to robot design, which tackles these aspects simultaneously.

Using ER techniques often requires evaluating the fitness function for many individuals over many generations, leading to a large number of evaluations. Employing ER methods for optimizing controllers on real robots tends to be expensive in terms of time consumption and mechanical wear. Some systems show successful evolution of a controller on a real robot, other studies are relying on simulations, where doing tens of thousands of evaluations is realistic and often necessary [Doncieux 2010] [Hettiarachchi 2012]. ER methods can be used to learn, or rather optimize, various controllers, for example, Central Pattern Generators [Habu 2019] on a simulated quadruped or a periodic function motion model for a bipedal robot [Maximo 2017].

In the case of morphology optimization, where the body of a robot is being evolved according to a fitness function, evaluating on a real robot poses additional challenge of constantly modifying the mechanical design of the robot. The robot would have to be disassembled and modified each time the morphology changes, which happens constantly with every new evaluated individual in the evolutionary process. This could be addressed by modular structures, either self-reconfigurable or manually configurable [Brunete 2017]. Self-reconfigurable modules can change the design of the whole robot themselves in between evaluations [Alattas 2019]. Manually reconfigurable modules require an operator to reconnect the modules into different configurations, which can be very time consuming for a researcher. Assembling modules by hand can be avoided by using an external robotic manipulator to put modules together, modules can automatically connect with magnets [Moreno 2018], or be glued together [Brodbeck 2015].

Modular robots can be relatively quickly rebuilt into different morphologies, although these morphologies will depend heavily on the size and capabilities of individual modules. Different approach is to have robots with links that can change shape. Robots with adaptive morphology [Mintchev 2016] could be used to optimize their bodies during their operation. Nygaard et al. demonstrated a quadruped robot with dynamic morphology [Nygaard 2018], where a robot can change the length of its legs automatically.

Even though testing the methods of evolutionary robotics on real robots has been done to some extent, still the majority of work in this area is done in simulations. Bringing solutions evolved in simulations into the real world often creates a problem referred to as reality-gap. Koos et al. [Koos 2013] marked reality-gap as arguably the most critical issue that currently prevents the use of ER for practical robotic applications and introduced a transferability approach to evaluate controllers. Nick Jakobi et al. [Jakobi 1995] showed that to get almost identical behavior in reality, appropriate levels of noise must be included in the simulation.

Optimizing a robot can also be done by optimizing its parts using more general engineering methodologies, namely structural topological optimization. This methodology is becoming increasingly important as it allows for higher performance, lightweight structures, and higher efficiency. [Zargham 2016]. Topological optimization gives answer to the placement of material within a prescribed design domain in order to obtain the best structural performance. The method has been developed in the 80s and has seen a lot of development since [Sigmund 2013]. There are many programs using the methods of topological optimizations available to engineers.

In this article, we compare two approaches to optimizing a quadruped robot. Topological optimization as an example of a more standardized approach from mechanical engineering and a more experimental approach in the form of a genetic algorithm. The second chapter describes the robot used for the optimization, its parameters and features. The third chapter covers topological optimization used to minimalize the weight of the robot. The fourth chapter describes our approach of using a genetic algorithm to optimize the morphology of the robot. This is followed by the results in chapter five where we show our measurements from simulations and laboratory tests.

2 QUADRUPED ROBOT

The robot used in this paper is a quadrupedal walking robot, developed by the Department of Robotics, VSB-TUO. It is a small robot, weighting about 1850 g. The body of the robot contains control electronics and a li-po accumulator. The robot and its dimensions can be seen in Figure 1.

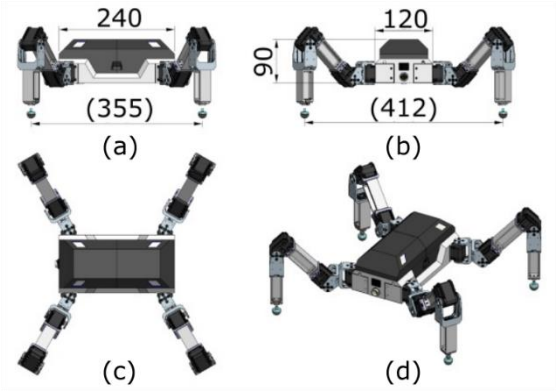


Figure 1. Robot dimensions; (a) side view, (b) back view, (c) top view, (d) isometric view

Parameter	Value
Body dimensions	240x120x90 mm
Weight	1850 g
Battery	Li-po 3S 1550 mAh 11,1 V
Powered time	Up to 50 minutes
Motors	12 x Dynamixel AX-12
Leg length (links)	379 mm (49 + 130 + 194)

Table 1. General parameters of the robot

2.1 Control

The whole system contains the quadrupedal robot and a PC, which the operator uses. The robot receives velocity commands $(\boldsymbol{v_X}, \boldsymbol{v_Y}, \boldsymbol{\omega})$ and other parameters from a PC and sends back sensor measurements. The robot then executes movements according to the set velocity. Gait generator and inverse kinematics are calculated in real-time onboard the robot and the resulting joint angles are set.

The robot runs on a microcontroller Netduino 2 Plus, which is programmed in .Net C\#. The microcontroller board is expanded with a custom-made module, often called a shield (Figure 2). This shield expands the capabilities of the Netduino platform with current measuring sensors, a buffer that allows control of Dynamixel servos and additional connectors. A Bluetooth-UART module is used for communicating with the PC. Each leg is actuated by three Dynamixel AX-12 servomotors.

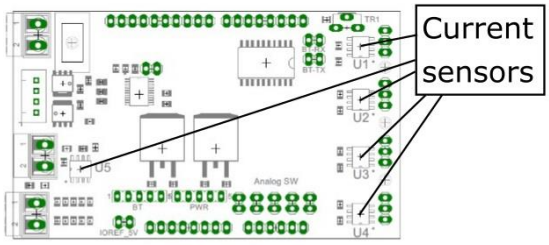


Figure 2. Sensor shield

2.2 Gait generation

In all tests in this article, the robot was walking in a static gait, where at least three legs are always in contact with the floor. The gait generator computes the pose of five points, namely, the robot body and tips of each leg. These points are moving on trajectories driven by parameters, such as velocity, body orientation, gait timing, and others. Calculated poses are then processed with inverse kinematics into joint variables.

The robot has 12 DOF (Figure 3), which can be divided into four serial linkages of three DOF. The inverse kinematics method used in this robot is a simple analytical method (equations 1-7), which maps the pose (X, Y, and Z, ignoring orientation) of the leg tip to the joint space $(\theta_1, \theta_2, \theta_3)$ (Figure 4).

$$\rho = \sqrt{x_P^2 + y_P^2} \tag{1}$$

$$c = \sqrt{z_P^2 + (\rho - l_1)^2} \tag{2}$$

$$\alpha = \operatorname{acos}\left(\frac{c^2 + l_2^2 - l_3^2}{2 \cdot c \cdot l_2}\right) \tag{3}$$

$$\gamma = \operatorname{acos}\left(\frac{l_3^2 + l_2^2 - c^2}{2 \cdot l_3 \cdot l_2}\right) \tag{4}$$

$$\psi = arctg\left(\frac{z_P^2}{(\rho - l_1)^2}\right) \tag{5}$$

where x_P, y_P, z_P is the end point of the leg, and l1,l2,l3 is the length of the first segment. Then the joint angles for the second and third joint are

$$\theta_2 = \psi + \alpha \tag{6}$$

$$\theta_3 = \gamma - \pi \tag{7}$$

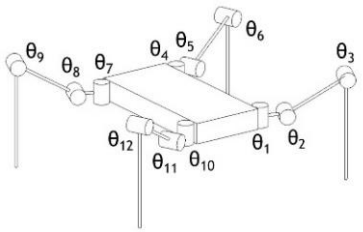


Figure 3. Degrees of freedom

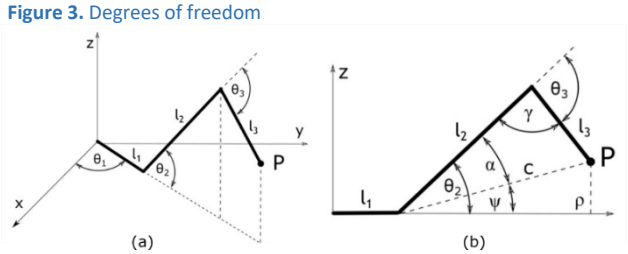


Figure 4. Leg kinematics. (a) In the first motor frame, (b) in the leg plane

2.3 Sensors

The robot is equipped with four Hall effect sensors ACS712, that monitor the current draw of each leg separately, and one Hall effect sensor ACS711, monitoring the current draw of the entire robot. Current measurements are sampled at 100 Hz and a moving average filter of 5 samples. An inertial measurement unit (UM7 from Redshift Labs) measures the orientation in space and accelerations exerted on the robot's body. Sensor data are transmitted to the operator and recorded at the rate of 15 Hz. Dynamixel AX-12 servos have a built-in functionality to measure the applied torque. Resolution of this torque measurement is however very low, and we did not use it.

2.4 Simulation

Dynamic simulations proved to be a very effective tool when designing a control system of the robot. The software used for simulations of this robot is V-rep by Coppelia Robotics. The dynamic simulation runs on a Newton physics engine, with a step-size of 50 ms.

The model has the same physical parameters as the real robot including kinematics, dimensions, weight, inertia and motor torques (Figure 5). This model server not only for testing the control system but is an important tool throughout the whole article. It is used to get force variables necessary for topological optimization in chapter 3. V-rep software has a remote API, which allows other programs to control various aspect of the simulation. Thanks to the API, it is able to serve as a means of evaluating the fitness in chapter 4. And finally, simulations of the optimized robots serve as a comparison to the real laboratory tests in chapter 5.

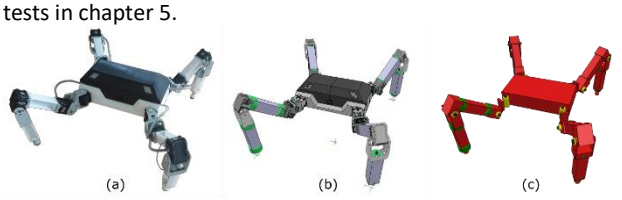


Figure 5. (a) Real robot, (b) visual simulation model, (c) dynamic simulation model

We can measure various parameters of the model during the dynamic simulation. The joints of the mechanism can measure their the applied torques. There is also a number of force-torque sensors embedded in each leg. Figure 6 shows the placement of force-torque sensors in the leg and numbering of the leg segments. The first and second segment are from both sides connected to a servo motor while the third segment is connected to a servo motor on one side and ends with a leg tip. In this paper we will focus on optimizing the second and third segment of the leg.

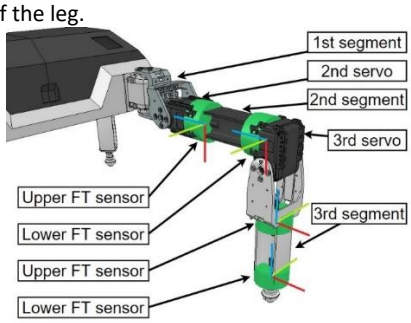


Figure 6. Force sensor placements in the simulation model

The force-torque sensors will be used to get force constrains for topological optimization in chapter 3. (Figure 7) Torque measurements will be used to compute fitness value for genetic algorithm in chapter 4. and for final comparison between the methods in chapter 5.

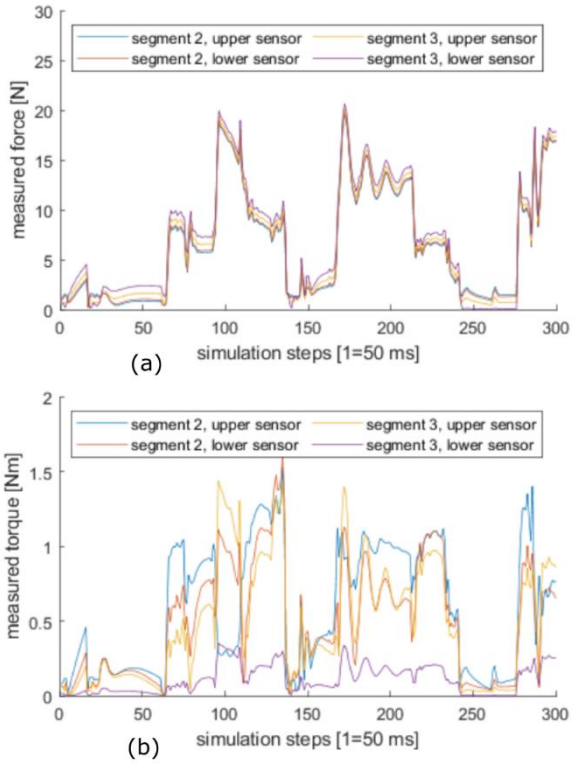


Figure 7. An example of measurements from FT sensors in the legs. Measured while walking in simulation: (a) Force measurements, (b) Torque measurements

3 TOPOLOGICAL OPTIMIZATION

In this chapter we describe a more traditional approach to optimizing mechanical components using topological optimization (TO). TO has been a part of dedicated programs for decades and in recent years has found its way into mainstream CAD software packages (SolidWorks, Fusion360, Catia and others). It is a proven approach in mechanical engineering with a wide user base. We are using it as an example of a more traditional approach and a benchmark against the more scientific approach in chapter 4.

Topological optimization uses a prior knowledge of loads and constrains acting on a mechanical part to minimize the used material. It uses a defined set of constrains to drive the optimization process. The constrain can be fixed surfaces or boundary conditions, that cannot change during the optimization. In our case the constrains are the connecting flanges on each end of the leg segment. The leg does not change length during the process, only the internal structure is modified. The kinematics and therefore the capabilities of the robot should stay the same while the efficiency should increase thanks to the removed weight.

To get a set of loads for the optimization, we used the simulation model described in the previous chapter. We simulated the robot while walking in various conditions and measured forces and torques in the leg sensors. One example of a measurement is shown in Figure 7.

The model was simulated with various settings, to determine and measure the loads in the legs during walking. The maximum measured loads from V-rep simulation were used as a parameter for the optimization in solidThinking Inspire software (Figure 8). The material used in optimization is PA 12 (PA 2200), with tensile strength of 48 MPa. The part was allowed displacement of 1,5 mm.

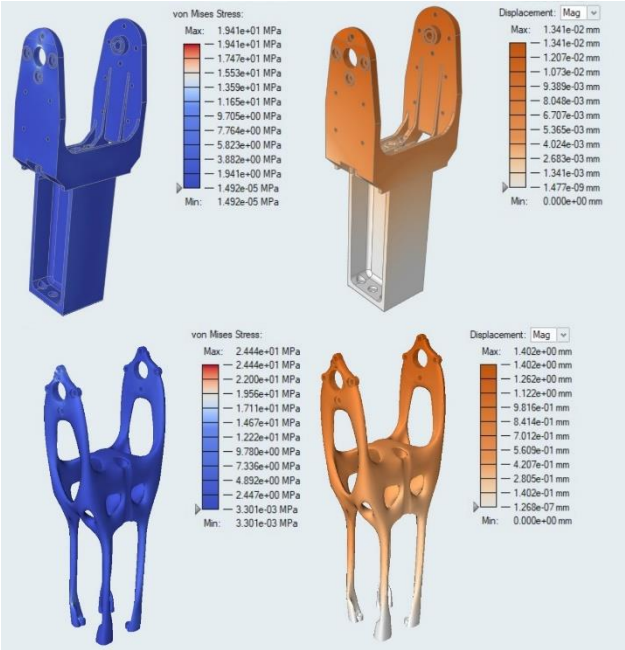


Figure 8. Segment in SolidThinking Inspire; (a) Von Mises stress before TO, (b) displacement before TO, (c) Von Mises stress after TO, (d) displacement after TO

Figure 9 shows the leg design after topological optimization. The new leg segments were printed out of ABS and PA 12, making the whole robot 142 g lighter (Table 2).

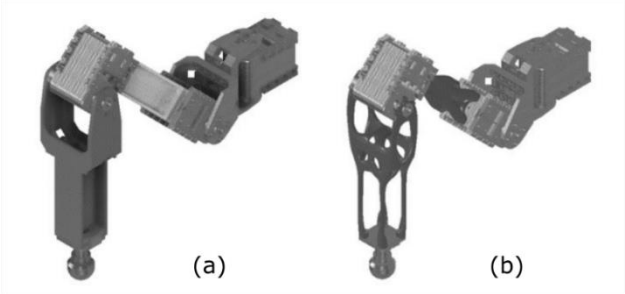


Figure 9. (a) Original leg, (b) Topologically optimized leg

Part	Original	Optimized
2nd segment	28.7 g	17 g
3rd segment	36.8 g	13 g

Table 2. Leg segment mass

4 GENETIC OPTIMIZATION

The use of genetic algorithms (GA) is very frequent in evolutionary robotics. Robots can evolve controllers for their locomotion, environment specific behaviors, their morphology, or multiple things at once. Several papers used GA to optimize the morphology of quadrupedal machines, mostly virtual creatures [Larpin 2011][Heinen 2009][Bongard 2011][Nygaard 2016][Leger 1999]. Even though the use of GA and more broadly methods of evolutionary computing have been around for decades we can still consider their use in everyday engineering as an experimental approach.

In this paper, we are using a GA to optimize the length of the leg segments. Each leg in the simulation model (Figure 6) has two segments that can change length before running the simulation. The inverse kinematics in our control script (chapter 2.2.) can handle changes in length of individual legs. We have however decided not to optimize all 8 leg segments individually and decided to use left-right symmetry. This way both front legs have segments of the same length. The same goes for back legs. By mirroring the mechanism, we are making the search space smaller with only 4 lengths to optimize instead of 8. We can also assume that symmetrical body will perform better during walking than an asymmetrical body.

We are using a GA implementation from Matlab optimization toolbox. The population size was set to 50. There were 5 input parameters for the optimization, the front and back leg segments and a velocity (Table 3). Fitness function was calculated in a control script that communicated with the simulation in V-rep and returned fitness value back to the GA. Figure 10 shows the overview of the connection between V-rep and Matlab.

Parameter	lb	ub	result
Front leg 2nd segment	0 m	0.5 m	0.050 m
Front leg 3rd segment	0 m	0.5 m	0.007 m
Rear leg 2nd segment	0 m	0.5 m	0.051 m
Rear leg 3rd segment	0 m	0.5 m	0.056 m
Set velocity	0 m/s	1 m/s	0.045 m/s

Table 3. Optimized parameters, their limits (lb = lower bound, up = upper bound) and the best individual.

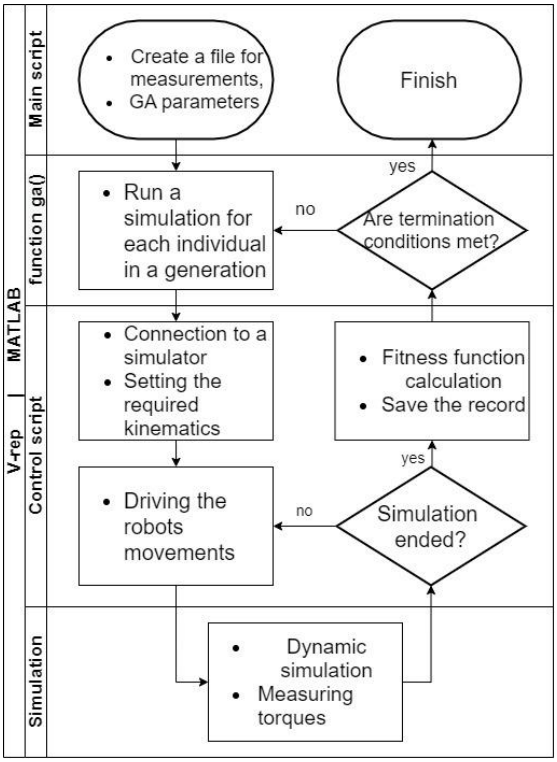


Figure 10. Diagram of the optimization

Matlab implementation of GA minimizes the value of an objective (fitness) function. Therefore, our fitness function is formulated to be minimized. After each simulation, the values of average torque on the motors and the traversed distance are requested from V-rep and a fitness is calculated according to the formula

$$fitness = \tau \cdot w - P_X \tag{8}$$

where τ is an average torque from the simulation, PX in the translated distance in the X axis and w is a multiplier, which puts the torque into the same range as the distance.

Ten runs of the GA optimization were performed, each taking about 30 hours. Figure 11 shows the evolutionary progress of the 10 runs. Fitness, torques and distance travelled are averaged and shown as the black line. The vertical error bars show the standard deviation.

The GA did in some cases abuse conditions of the simulation. In those cases, the robot had small front legs and long rear legs, which caused the robot to flip forwards, and thus move in the X direction and remain on its back with the legs moving through air with small torque on the motors. This made the GA stuck in a local minimum.

In other cases, the GA produced successful kinematics. The best individual from GA optimization has shorter legs, the 3rd segment of the front legs is shortened to mere 7 mm (Table 2). The segments were 3d printed and installed on the robot for laboratory testing.

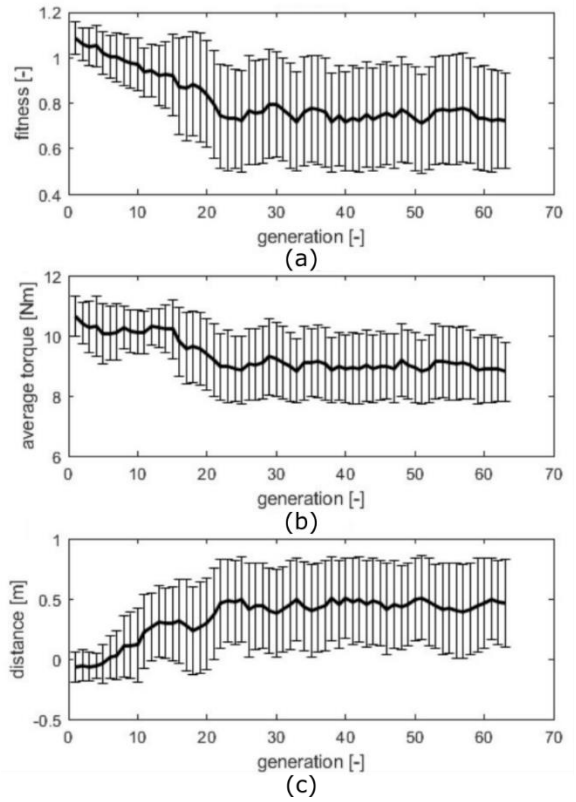


Figure 11. Average progress of the genetic algorithm; (a) Fitness, (b) Torque ,(c) Traveled distance

5 EXPERIMENTS

To test how the improvements in current consumption compare between simulation and reality, we tested the robot walking in the same conditions in our lab. All three versions of the robot (original, GA kinematics, and topologically optimized) were simulated in V-rep and tested in our lab.

Parameter	Value
Set velocity	Forward 40 mm/s
Body height	150 mm
Cycle time	3 s
Leg-swing height	50 mm

Table 4. Velocity and posture parameters for test

5.1 Simulation

The simulation here is very similar to the one used during the genetic algorithm optimization, described above. Unlike the simulation during the GA optimization however, we are measuring the torque for all the time steps of the simulation, instead just the final average. Each simulation is run for 500 simulation steps, which corresponds to 25 seconds of simulation time. The simulations are repeated and recorded ten times for each model.

5.2 Lab setup

The robot was placed on an even floor under a camera (Figure 12). The body of the robot is equipped with an ArUco marker and the camera detects the position of the marker. The camera is an Intel Realsense D435i, which is an RGBD camera, however only RGB images were used in detecting the position of the robot. The detection was done using OpenCV and the ArUco module. The detection was done on pictures at resolution of 1280x720 pixels.

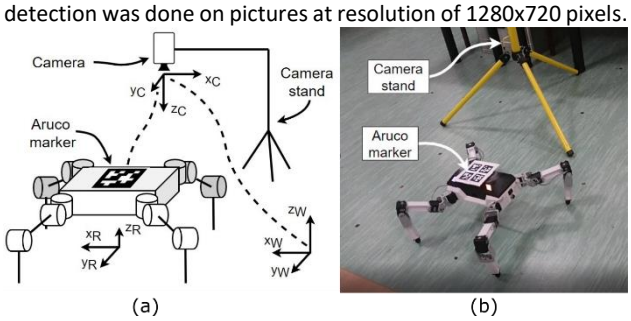


Figure 12. Lab setup; (a) Schematic, (b) photo from the lab.

5.3 Measuring

The robot walked under the camera for 25 s, which is the same time we used in our simulations. Electrical current measurements were recorded during the walking. This walking test was repeated and recorded ten times. Then we rebuilt the robot to a different configuration and tested the same way again. Figure 13 shows the three configurations on the laboratory floor.

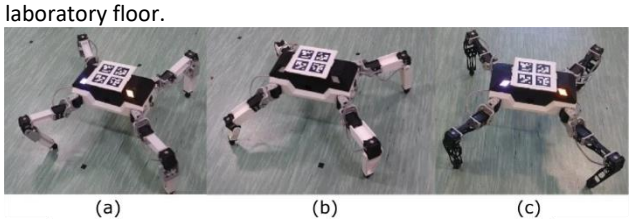


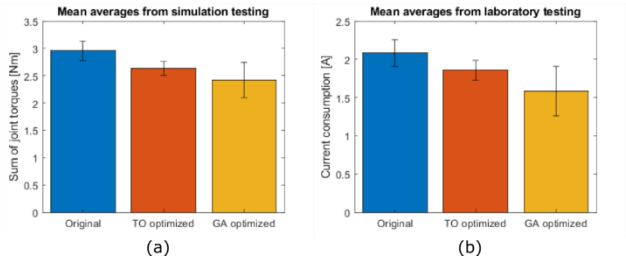
Figure 13. Three different configurations during the walking test (a) original robot, (b) GA optimized morphology, (c) Topologically optimized legs

6 RESULTS

Is this section we examine the torque measurements from simulation and the electrical current measurements from the tests with real robots. We will not directly map the motor torque to current consumption, or vice-versa, as this would not be an accurate mapping given the used sensors. Instead we will consider both of these measurements as the measurement of the robot efficiency and look at the percentage of its improvement.

Each bar in Figure 14 shows the average of ten tests with one configuration. We can see that the genetic algorithm achieved better results than topologic optimization in both simulation and laboratory testing.

The results are more apparent in Table 5. Third and fifth column show percentage improvement of an optimized configuration relative to the original configuration. The last column shows the difference between improvement in simulation and improvement in laboratory testing.



(a) (b)

Figure 14. Measured efficiency of the different robots. (a) Simulated test, (b) real test

Robot configuration	Simulation result [Nm]	Improvement in simulation [%]	Lab experiment [A]	Improvement on real robot [%]
Original configuration	2.96	-	2.03	-
Topologically optimized	2.63	11.0	1.86	10.7
GA optimized morphology	2.42	18.2	1.58	23.9

Table 5. Experimental results

One of our conditions in optimizing the robot efficiency was to keep performance. When we investigate the trajectory that the robot body creates, we can see a significant difference between simulation and lab testing. Figure 15 shows how much does the robot body deviate from the simulated trajectory. For clarity, only one simulated trajectory is shown since they are very similar. The change in the overall direction is most likely caused by the robot slipping on the floor, as it seems that the trajectory randomly shifts to the left and right over the several repetitions. Another difference from the simulation is the amplitude of the swinging motion, that the robot body performs to keep its center of mass above feet with floor contact.

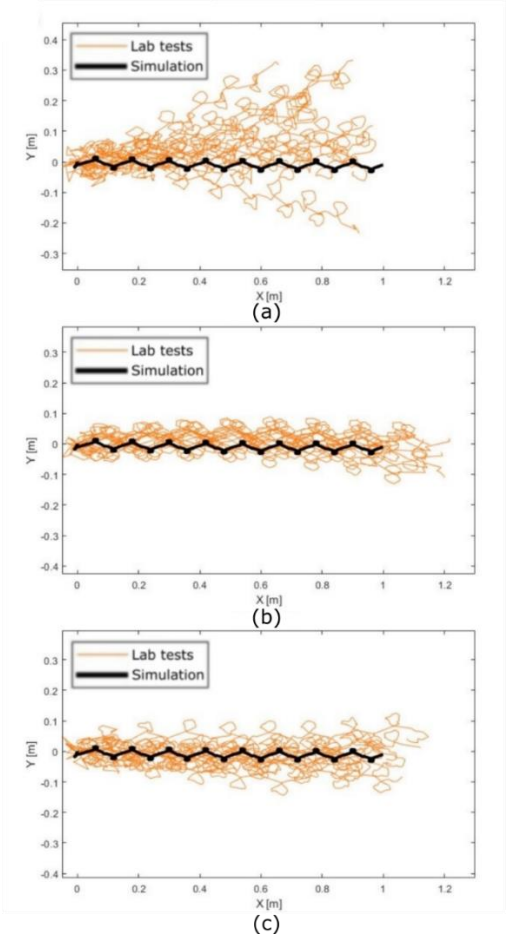


Figure 15. Position of the robot during walking (XY plane - floor), (a) original robot, (b) GA optimized kinematics, (c) Topologically optimized design

7 RESULTS

A selected quadrupedal robot was described and then modelled in simulation software V-rep. Two approaches for optimizing efficiency of the robot were tested and compared. The first approach was topological optimization which is more traditional and something a mechanical engineer might use. As a second approach we used a genetic algorithm which we estimate would be preferred by a computer scientist. In both cases the subject of optimization was the length of the leg segments.

We have tested the original robot and the optimized robots in simulation and obtained improved results. The robot optimized with TO showed an improvement of 11% and the robot optimized with GA showed an improvement of 18.2%. Then we have tested the robots in real life laboratory testing. In real life test the robot optimized with TO showed an improvement of 10.7%, which was very similar to the improvement in simulation. The robot optimized with GA showed 23.9% improvement. This means that it performed 5.7% better than what we would expect from our simulation. The discrepancy, often called reality gap, could be caused by a number of things. The GA changed the lengths of robot's legs therefore it might affect the behavior of the friction in robot's joints and the slippage of the legs. These things are difficult to approximate in simulation.

An interesting behavior was revealed when measuring exact trajectories of the robot body with a motion capture system from an overhead camera. Both TO and GA optimized robots show a lot less disturbance when walking in a straight line when compared to the original robot. This is not something that we expected since we did not optimize for straight walking but for efficiency which was measured as a sum of motor torques. It might be that optimizing for efficiency intrinsically makes other features of the robot behavior better.

Naturally when both approaches show an improvement, why not use both GA and TO? Using both GA with a full dynamic model and a TO at the same time is problematic. Mainly because in our case the GA uses prior knowledge of density of the leg segments (the weight of a segment is linearly dependent on its length) and TO changes the density. At the same time, TO uses prior knowledge of the forces, torques and constrains for each mechanical part and because GA changes the length of the leg segments during its run it would be in effect changing the constrains for TO. Their use at the same time is problematic because they would change each other initial conditions. Possible solution would be to run each of them in turns and iterate towards an optimal morphology and topology this way. The authors are not aware of a project that would combine these two methods.

Our future work will focus on using similar methods of optimization on modular manipulators. Minimizing the measured differences between simulation and reality by dynamically refining the simulated model during optimization.

8 CONCLUSIONS

We could conclude that optimizing a robot with topological optimization is more predictable than optimizing it with genetic algorithm. Optimizing the whole robot morphology with a genetic algorithm has showed better improvements than optimizing the legs with topology optimization. It is hard to say that modifying a robot's morphology with a GA is a generally better approach than optimizing its parts with TO. We had the opportunity to change the morphology without changing the

robot's performance, this is often not the case with robots in practical use.

Optimizing with TO is a rather standard process these days, it however creates machine parts that are difficult to manufacture and therefore is not suitable for wide range of uses due to the manufacturing costs. On the other hand, optimizing morphology with GA is a lengthy process usually with a lot of custom solutions. The resulting morphology from GA, however, can be manufactured with more conventional methods. It could be therefore thought of as a tradeoff between design and manufacturing phases.

ACKNOWLEDGMENTS

This work was supported by the European Regional Development Fund in the Research Centre of Advanced Mechatronic Systems project, project number CZ.02.1.01/0.0/0.0/16_019/0000867 within the Operational Programme Research, Development and Education.

This article has been completed in connection with project Innovative and additive manufacturing technology – new technological solutions for 3D printing of metals and composite materials, reg. no. CZ.02.1.01/0.0/0.0/17_049/0008407 financed by Structural Funds of European Union and project.

This article has been also supported by specific research project SP2020/141 and financed by the state budget of the Czech Republic.

REFERENCES

[Alattas 2019] Alattas RJ, Patel S, Sobh TM. Evolutionary Modular Robotics: Survey and Analysis. J Intell Robot Syst Theory Appl [Internet]. 2019 Sep 14 [cited 2020 Nov 6];95(3–4):815–28. Available from: https://doi.org/10.1007/s10846-018-0902-9

[Bongard 2011] Bongard J. Morphological change in machines accelerates the evolution of robust behavior. Proc Natl Acad Sci U S A [Internet]. 2011 Jan 25 [cited 2020 Nov 6];108(4):1234–9. Available from: www.pnas.org/cgi/doi/10.1073/pnas.1015390108

[Brodbeck 2015] Brodbeck L, Hauser S, Iida F. Morphological Evolution of Physical Robots through Model-Free Phenotype Development. Bongard J, editor. PLoS One [Internet]. 2015 Jun 19 [cited 2020 Nov 6];10(6):e0128444. Available from: https://dx.plos.org/10.1371/journal.pone.0128444

[Brunete 2017] Brunete A, Ranganath A, Segovia S, de Frutos JP, Hernando M, Gambao E. Current trends in reconfigurable modular robots design. Int J Adv Robot Syst [Internet]. 2017 May 12 [cited 2020 Nov 6];14(3):172988141771045. Available from: http://journals.sagepub.com/doi/10.1177/1729881417710457

[Doncieux 2015] Doncieux S, Bredeche N, Mouret JB, (Gusz) Eiben AE. Evolutionary robotics: What, why, and where to [Internet]. Vol. 2, Frontiers Robotics AI. Frontiers Media S.A.; 2015 [cited 2020 Sep 17]. p. 4. Available from: www.frontiersin.org

[Doncieux 2010] Doncieux S, Mouret JB. Behavioral diversity measures for evolutionary robotics. In: 2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010. 2010.

[Habu 2019] Habu Y, Uta K, Fukuoka Y. Three-dimensional walking of a simulated muscle-driven quadruped robot with neuromorphic two-level central pattern generators. Int J Adv Robot Syst. 2019;

[Heinen 2009] Heinen MR, Osório FS. Evolving morphologies and gaits of physically realistic simulated robots. In: Proceedings of the ACM Symposium on Applied Computing [Internet]. New York, New York, USA: ACM Press; 2009 [cited 2020 Nov 6]. p. 1161–5. Available from: http://portal.acm.org/citation.cfm?doid=1529282.1529540

[Hettiarachchi 2012] Hettiarachchi DS, Iba H. An Evolutionary Computational Approach to Humanoid Motion Planning. Int J Adv Robot Syst [Internet]. 2012 Nov 15 [cited 2020 Nov 6];9(5):167. Available from: http://journals.sagepub.com/doi/10.5772/51905

[Jakobi 1995] Jakobi N, Husbands P, Harvey I. Noise and the reality gap: The use of simulation in evolutionary robotics. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 1995.

[Koos 2013] Koos S, Mouret JB, Doncieux S. The transferability approach: Crossing the reality gap in evolutionary robotics. IEEE Trans Evol Comput. 2013;17(1):122–45.

[Larpin 2011] Larpin K, Pouya S, Van Den Kieboom J, Ijspeert AJ. Co-evolution of morphology and control of virtual legged robots for a steering task. In: 2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011. 2011. p. 2799–804.

[Leger 1999] Leger PC. Automated synthesis and optimization of robot configurations: An evolutionary approach. Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1999. Avaliable from: https://www.researchgate.net/publication/2595880_Automated_Synthesis_and_Optimization_of_Robot_Configurations_An_Evolutionary_Approach

[Maximo 2017] Maximo MR, Colombini EL, Ribeiro CH. Stable and fast model-free walk with arms movement for humanoid robots. Int J Adv Robot Syst [Internet]. 2017 May 14 [cited 2020 Nov 6];14(3):172988141667513. Available from: http://journals.sagepub.com/doi/10.1177/1729881416675135

[Mintchev 2016] Mintchev S, Floreano D. Adaptive morphology: A design principle for multimodal and multifunctional robots. IEEE Robot Autom Mag. 2016 Sep 1;23(3):42–54.

[Moreno 2018] Moreno R, Veenstra F, Silvera D, Franco J, Gracia O, Cordoba E, et al. Automated Reconfiguration of Modular Robots Using Robot Manipulators. In: Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018. Institute of Electrical and Electronics Engineers Inc.; 2019. p. 884–91.

[Nygaard 2018] Nygaard TF, Martin CP, Torresen J, Glette K. Exploring Mechanically Self-Reconfiguring Robots for Autonomous Design. 2018 May 8 [cited 2020 Nov 6]; Available from: http://arxiv.org/abs/1805.02965

[Nygaard 2016] Nygaard TF, Torresen J, Glette K. Multi-objective evolution of fast and stable gaits on a physical quadruped robotic platform. In: 2016 IEEE

Symposium Series on Computational Intelligence, SSCI 2016. Institute of Electrical and Electronics Engineers Inc.; 2017.

[Sigmund 2013] Sigmund O, Maute K. Topology optimization approaches: A comparative review [Internet]. Vol. 48, Structural and Multidisciplinary Optimization. Springer; 2013 [cited 2020 Oct 8]. p. 1031–55. Available from: https://link.springer.com/article/10.1007/s00158-013-0978-6

[Zargham 2016] Zargham S, Ward TA, Ramli R, Badruddin IA. Topology optimization: a review for structural designs under vibration problems. Struct Multidiscip Optim [Internet]. 2016 Jun 1 [cited 2020 Oct 8];53(6):1157–77. Available from: https://link.springer.com/article/10.1007/s00158-015-1370-5

CONTACTS:

Ing. Robert Pastor
VSB - Technical University of Ostrava, Department of Robotics
17. listopadu 2172/15, Ostrava, 708 00, Czech Republic
+420 597 324 125, robert.pastor@vsb.cz