Research Collection

Doctoral Thesis

Computational Design Synthesis for the Co-Design of Morphology and Actuation of Soft Robots for Locomotion

Author(s):
Van Diepen, Merel

Publication Date:
2020

Permanent Link:
https://doi.org/10.3929/ethz-b-000452781 →

Rights / License:
In Copyright - Non-Commercial Use Permitted →

This page was generated automatically upon download from the ETH Zurich Research Collection. For more information please consult the Terms of use.

ETH Library

DISS. ETH NO. 26879

COMPUTATIONAL DESIGN SYNTHESIS FOR THE CO-DESIGN OF MORPHOLOGY AND ACTUATION OF SOFT ROBOTS FOR LOCOMOTION

A dissertation submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

MEREL DIRKJE MARIE VAN DIEPEN
MSc Delft Technical University ME

born on 23 June 1983
citizen of the Netherlands

submitted to

Prof. Dr. Kristina Shea, examiner
Prof. Dr. Stelian Coros, co-examiner
Dr. Danny Kaufman, co-examiner
Asst. Prof. Dr. Christopher McComb, co-examiner

2020

Merel Dirkje Marie van Diepen: *Computational Design Synthesis for the Co-Design of Morphology and Actuation of Soft Robots for Locomotion,* © 2020

Compiled on July 28, 2020

ABSTRACT

Compliant robots, i.e. robots that yield to external forces, have advantages compared to rigid robots in several respects. In direct contact with humans, compliant robots are safer. They can also compensate for imperfect perception of the environment. For example, a compliant gripper can achieve an even force distribution when holding a tomato without knowing the exact shape of the tomato.

Soft robots are compliant robots that achieve compliant behavior through the use of compliant materials for their bodies. The field of soft robotic research is relatively young and next to the fabrication of soft robots, the design of soft robot morphology and actuation is still a challenge. Due to the vast design freedom, selecting a good design concept is challenging. Designing soft robots by hand often requires a trial-and-error process that is time-consuming. Thus, automation of the design can be advantageous. Currently, the majority of soft robot concepts are bio-inspired. Soft robots, however, can also be systems that are infeasible with biological systems. There is a need for a method to explore the solution space and enable an informed choice among concepts.

This work focuses on the task of locomotion since it is an essential component of common robotic tasks such as search and rescue, exploration and pick and place. The objective of this research is to develop a computational design synthesis method for the conceptual design of the combined morphology and actuation of soft robots for locomotion.

A Computational Design Synthesis (CDS) approach is taken in this thesis. It is an iterative process that generates, evaluates and optimizes design candidates. First, a spatial grammar for the generation of morphologies is presented. It can guide the process towards valid designs, since it enables fabrication requirements and constraints to be incorporated directly while still maintaining enough design freedom. The designs are evaluated in simulation using a rigid-body model for the approximation of soft robots and simulated annealing is used to optimize the designs.

Both the CDS of soft robot morphology and the CDS of morphology and actuation are presented. Designs generated with the CDS of morphology only are actuated by fixed actuation patterns. To optimize the actuation patterns for every individual design, Parametric Actuation Curves (PAC) optimization is formulated and integrated in the morphology evaluation phase. The resulting CDS method for the co-design of morphology and actuation is shown to be more effective and more efficient than the CDS method for the automated design of morphology only.

The designs resulting from the CDS method of morphology and actuation are prototyped successfully using two actuation methods; pneumatic actuation and tendon actuation. The different constraints and requirements of the two types of robots are incorporated directly in the spatial grammar. The prototypes compete with state-of-the-art soft robots with respect to speed and outperform them with respect to novelty, as defined by a novelty metric. The designs resulting from the presented co-design CDS method span a wide range of morphologies and gaits, many without biological analogy. Multiple solutions are generated with the same objective function value showing the multi-modality of the solution space. This work shows that the CDS of morphology and actuation enables a wider exploration of the entire solution space compared to bio-inspired design alone.

ZUSAMMENFASSUNG

Nachgiebige Roboter, d.h. Roboter, die externen Kräften bis zu einem gewissen Grad nachgeben, haben gegenüber starren Robotern Vorteile in verschiedener Hinsicht. Bei direktem Kontakt zwischen Mensch und Roboter sind nachgiebige Roboter sicherer. Ausserdem können nachgiebige Roboter Ungenauigkeiten in der Wahrnehmung der Umgebung kompensieren, z.B. kann ein nachgiebiger Greifer ohne die genaue Form einer Tomate zu kennen diese mit einer ausgeglichenen Kräfteverteilung halten.

Weiche Roboter sind nachgiebige Roboter, die ihre Nachgiebigkeit dem dehnbaren Material, aus dem sie hergestellt sind, verdanken. Das Forschungsgebiet der weichen Roboter ist noch ziemlich jung und neben den Herausforderungen im Bereich der Fertigung ist auch das Entwerfen weicher Roboter eine Herausforderung. Sich auf einen Entwurf festzulegen ist komplex, da das Verhalten des dehnbaren Materials schwer vorherzusagen ist und die Entwurfsfreiheiten riesig sind. Manuelles Entwerfen ist ein aufwändiger und zeitraubender Prozess, der auf Versuch und Irrtum basiert. Daher ist eine Vielfalt der Roboterkonzepte biologisch inspiriert. Allerdings ist es möglich Roboter zu realisieren, die mit biologischen Komponenten nicht möglich wären. Ausserdem gibt es keine zuverlässigen Methoden um verschiedene Konzepte zu vergleichen. In dieser Dissertation wird eine alternative Methode zum Entwurf von weichen Fortbewegungsrobotern präsentiert. Die Aufgabe sich fortzubewegen ist ein essentieller Teil von verschiedenen übergeordneten Roboteraufgaben, z.B. suchen und retten, explorieren sowie greifen und platzieren. Das Ziel dieser Dissertation ist die Entwicklung einer computergestützten Entwurfs-Synthese-Methode für den konzeptuellen Entwurf von Morphologie und Steuerung weicher Fortbewegungsroboter.

Eine Computergestützte Entwurfs-Synthese (CES) ist ein iterativer Prozess zum Generieren, Bewerten und Optimieren von Entwurfskandidaten. Erstens wird eine räumliche Grammatik für die Erzeugung von Morphologien präsentiert. Sie führt den Prozess in Richtung gültiger Entwürfe, beinhaltet Anforderungen und Einschränkungen auf eine direkte Weise und behält trotzdem genügend Entwurfsfreiheit. Zweitens wird ein Starrkörpermodel zur annähernden Simulation von weichen Robotern präsentiert. Während der Simulation werden die Aktuatoren mittels unveränderlicher Muster gesteuert. Die Roboter werden auf der Basis ihrer Leistung in der Simulation bewertet. Drittens werden die Roboterentwürfe mittels simulated annealing, einer metaheuristischen Optimierungsmethode, verbessert. Die resultierenden Entwürfe zeigen grosse Unterschiede in den Gangmustern. Weil ein nichtvariables Aktivierungsmuster den Suchraum einschränkt, wird die CES um die Optimierung eines Aktivierungsmusters für jeden einzelnen Entwurfskandidaten erweitert. Verschiedene Methoden zur Optimierung von Aktivierungsmustern werden geprüft und beurteilt auf Rechenaufwand, die Notwendigkeit Optimierungsparameter einstellen zu müssen und die Leistung der entworfenen Roboter. Die ausgewählte Steuerungsstrategie ist Parametrische

Aktivierungskurven (PA), eine feedforward Strategie, welche auf der Annahme basiert, dass die Steuerung zyklisch ist. Die PA werden optimiert für jeden zu evaluierenden Entwurf. Die resultierende CES Methode für das parallele Entwerfen von Morphologie und Aktivierung übertrifft die CES Methode für das alleinige Entwerfen der Morphologie.

Schliesslich werden die aus der CES Methode resultierenden Entwürfe mit zwei unterschiedlichen Aktuatorentypen gefertigt; pneumatische Aktuatoren und Seilzugaktuatoren. Die unterschiedlichen Bedingungen und Einschränkungen, die aus den Aktuationsmethoden folgen, können mit wenig Aufwand in die CES Methode eingebaut werden. Die entstandenen Prototypen sind vergleichbar mit den modernsten weichen Robotern was ihre Geschwindigkeit betrifft und übertreffen diese im Neuheitsgrad. Obwohl die Prototypen die durch die Simulation vorhergesagte Geschwindigkeit nicht ganz erreichen, genügt die Präzision der Simulation für konzeptuelles Entwerfen, und potenzielle Lösungen für die Steigerung der Präzision werden diskutiert.

Zusammenfassend bietet die hier präsentierte generative Entwurfsmethode eine Alternative zum biologisch inspirierten Entwerfen von weichen Fortbewegungsrobotern, fördert den Vergleich von Entwurfskonzepten und macht es möglich Entwürfe für verschiedene Aktuations- und Herstellungsmethoden zu generieren.

ACKNOWLEDGEMENTS

I would like to thank the people that made the last years productive and fun.

My supervisor Professor Dr. Kristina Shea I thank for giving me the opportunity to learn from her and always pushing me to become a better researcher. The EDAC-gang I thank for the many coffees, parties, road trips and snow days and for being brilliant sparring partners, mental battles only of course.

Els and Gregoor I thank for being awesome and sending me good vibes from the flatlands. I thank Paul for being everywhere I go.

My Zürich friends I thank for all the sporty and sometimes alcoholic times and my Dutch ones for disproving 'out of sight, out of mind'. The kickboxers at the Power Sports Club I would like to thank for punching some sanity back into my skull on a regular basis. To my flatmates I am grateful for making the house a real home with Limat swimming, pancakes, hiking and pj Sundays.

Also, I thank sugar and loud music for their everlasting support. Koodo's to ya all.

CONTENTS

ABSTRACT iii
ZUSAMMENFASSUNG v
ACKNOWLEDGEMENTS vii
LIST OF FIGURES xii
LIST OF TABLES xix

PREFACE xxi

1 INTRODUCTION 1
- 1.1 Scope of the Thesis 4
- 1.2 Expected Contributions 5
- 1.3 Design Task Formulation 6
- 1.4 The Computational Design Synthesis Methodology 6
- 1.5 Dissertation Overview 8

2 BACKGROUND 11
- 2.1 Related Work on Soft Robots 11
  - 2.1.1 Soft Robot Design 11
  - 2.1.2 Soft Robot Fabrication 12
  - 2.1.3 Soft Robot Control 12
- 2.2 Related Work on Design Automation 13
  - 2.2.1 Design Generation Methods 14
  - 2.2.2 Evaluation Methods 16
  - 2.2.3 Optimization Methods 17
- 2.3 Related Work on Control Optimization within Automated Design 18
- 2.4 Summary 19

3 SIMULATING SOFT BODIES 21
- 3.1 Finite Element Method 22
- 3.2 Soft-body Model 23
  - 3.2.1 Force-Based Spring-Mass Models 23
  - 3.2.2 Position-Based Soft-Body Dynamics 24
- 3.3 Rigid Body Model of Soft Body Simulation 24
  - 3.3.1 Rigid Ball-Spring Model 25
  - 3.3.2 Rigid-Body-Hinge Model 25
- 3.4 Discussion 27
- 3.5 Summary 28

4 COMPUTATIONAL DESIGN SYNTHESIS OF SOFT ROBOT MORPHOLOGIES FOR LOCOMOTION 29
4.1 Evaluation and Optimization 29
4.2 Study 1 31
4.2.1 Spatial Grammar 1 31
4.2.2 Results 36
4.2.3 Grammar Performance 38
4.2.4 Discussion Study 1 40
4.3 Study 2 42
4.3.1 Spatial Grammar 2 43
4.3.2 Optimization 46
4.3.3 Results 46
4.3.4 Grammar Performance 52
4.3.5 Discussion Study 2 54
4.4 Summary 54
5 COMPUTATIONAL DESIGN SYNTHESIS OF MORPHOLOGY AND ACTUATION OF SOFT ROBOTS FOR LOCOMOTION 55
5.1 CDS with Control as a Constraint 56
5.2 Finding a Suitable Actuation Optimization Method 59
5.2.1 Parametric Actuation Curve Optimization 59
5.2.2 Reinforcement Learning 60
5.2.3 Comparison of the Methods 64
5.3 CDS with PAC Optimization 66
5.4 Generating an Existing Design 72
5.5 Discussion 74
5.6 Summary 75
6 PROTOTYPING GENERATED SOFT ROBOT DESIGNS 77
6.1 Prototyping Methods 77
6.1.1 Prototypes using Pneumatic Actuation 77
6.1.2 Prototypes using Tendon Actuation 79
6.2 Design Generation 81
6.2.1 Modifications for Pneumatic Actuation 83
6.2.2 Modifications for Tendon Actuation 84
6.3 Results 85
6.3.1 Comparing the Prototypes to State-of-the-Art Soft Robots 88
6.4 Discussion 90
6.5 Summary 91
7 DISCUSSION 93
7.1 RQ 1.1: Morphology Generation 94
7.2 RQ 1.2: Simulation of Soft Robots 95
7.3 RQ 1.3: Optimization 96

7.4 RQ 2.1 and 2.2: Co-design of Morphology and Actuation 97
7.5 RQ 3.1 and 3.2: Prototyping 98
7.6 Thesis Objective 99
8 CONCLUSION AND FUTURE WORK 103
A SOFT-BODY DYNAMICS 107
A.1 Force-Based Soft-Body Dynamics 107
A.2 Position-Based Soft-Body Dynamics 108
B DEEP Q-NETWORKS REINFORCEMENT LEARNING 113
C NOVELTY OF STATE-OF-THE-ART SOFT ROBOTS 117

BIBLIOGRAPHY 119

CURRICULUM VITAE 127
PUBLICATIONS 129

LIST OF FIGURES

Figure 1.1 An example of a soft gripper for food handling, from [15]. 2

Figure 1.2 Several examples of soft robots for locomotion. a) A tetrapod actuated by shape memory alloys [21], b) a snake-bot using fluidic elastomer actuators [22], c) a pneumatic jumping robot [23] and pneumatic tetrapod for walking [10]. 3

Figure 1.3 Soft robot designs resulting from automated design processes. a) Several of the designs resulting from the computational design synthesis of soft robots from [27], b) the two results from the computational design sysnthesis reported by [28]. Comparing these designs to the designs in Figure 1.2, the lack of extremities stands out. 4

Figure 1.4 A robotic design is used to illustrate the two objectives of the optimization. The first objective, f_1, is the shortest distance traversed by any point of the design in the horizontal plane and is to be maximized. The second objective, f_2, is the average lost potential energy of the elements of the design and this objective is to be minimized. . 6

Figure 1.5 Overview of the computational design synthesis approach with and without design specific control. 7

Figure 1.6 Thesis roadmap, including where the research questions are answered. 9

Figure 1.7 An overview of the iterative design process, including the chapters further details on the design stages can be found. 10

Figure 2.1 An illustrative example of a spatial grammar with one rule. The rule replaces a circle, the left hand side, with a rectangle, the right hand side. The width of the rectangle is dependent on the parameter X. An rule application example with $X = 5$ is shown at the bottom. . . 16

Figure 3.1 (left) A physical inflatable bending actuator with a partial overlay of an FEM simulation, (middle) the same actuator modeled by rows of balls interconnected with springs, (right) the same actuator modeled using hinge constraints. 22

Figure 3.2 FEM simulation of an inflating bending actuator as part of the bachelor thesis conducted by Michael Dreyer (2016) and supervised by the author. 23

Figure 3.3 The spring-mass model on the top uses point-masses connected by distance constraints to represent elastic material. At the bottom, a model is shown where the point-masses are replaced by rigid bodies. 25

Figure 3.4 a) Example of a robot modeled with a ball-spring model. b) A ball and its 26 neighbors, of which 18 are direct neighbors (in orange). Only the direct neighbors are connected with distance constraints (black lines). 26
Figure 3.5 By decreasing the springs rest-lengths and the ball size in the bottom row, and increasing them in the top row, bending is achieved. 26
Figure 3.6 The bending actuator used as a model in the CDS. a) A pneumatic actuator. b) A shape memory alloy bending actuator resulting from the bachelor thesis of Matteo Tricarico, ETHZ, 2017 [81]. 27
Figure 3.7 Example of a robot modeled with a body-hinge model. 27
Figure 4.1 The four patterns used for activation. 31
Figure 4.2 Bending is achieved by increasing the length of the springs between balls on one side, here the top row, and decreasing the rest length on the other side, i.e. the bottom row. In simulation, the size of the balls is changed accordingly to keep balls from one row passing through the gaps of the neighboring row. 32
Figure 4.3 The seven rules of the ruleset. The first column shows the left hand side, the second column the right hand side. The third column shows the required parameters and their range. Blue balls (found in the extremities of rule 1, 2 and 4) have an increased friction coefficient. Bending arrows show the bending direction of the actuators. 34
Figure 4.4 a) The generation of a design created by hand-picking and applying rules. For every application, the rule and the used parameters are given. b) The motion and objective function value of the hand-built design. 35
Figure 4.5 A flowchart of the optimization loop. 35
Figure 4.6 The average convergence and standard deviation of 31 runs of 150 iterations with 50 moves. 36
Figure 4.7 The average convergence and standard deviation of 40 runs of 100 iterations with 25 moves. 36
Figure 4.8 Symmetric designs showing hopping and crawling locomotion. Design A crawls using the impulse from throwing its backside extension up. Design B uses the bending of its back vertical extension to hop forward. Design C hops forward by lifting off and rotating before landing. 38
Figure 4.9 Symmetric designs showing legged locomotion. By lifting and bending one side at the time and using the opposite side as pivot point, Design A moves forward. Design B pushes itself forward with its alternating back legs (dark blue extensions). Design C has a four-legged gait. 39

Figure 4.10 Asymmetric designs showing legged, hopping and crawling locomotion. Design A hops. Design B crawls with a motion that resembles the motion of a caterpillar. Design C shows two-legged locomotion (the two dark blue parts are the legs) and has a dynamic gait. Design D uses symmetric diagonal legs. Here, the symmetry evolved naturally and was not enforced during design generation. 40
Figure 4.11 Average improvement of the objective value for the different rules. All rules have, on average, a negative effect on the objective value, but a small percentage of rule applications improve the design. The median is shown as a line, the standard deviation as a box. 41
Figure 4.12 The seven rules of the rule set. Rules 0 to 5 add material, Rule 6 removes material. The rule parameters are listed in the right column, the activation directions are shown by the arrows. Dark blue balls have a higher friction value than the light grey balls. 44
Figure 4.13 a) The generation of a design created by hand-picking and applying rules. For every application, the rule and the used parameters are given. b) The motion and objective function value of the hand-built design. 45
Figure 4.14 The average accepted objective function value and the average best objective function value of 24 runs with 150 iteration and 50 moves with reheating. The standard deviation of the average accepted value is shown. 48
Figure 4.15 The average accepted objective function value and the average best objective function value of 24 runs with 150 iteration and 50 moves without reheating. The standard deviation of the average accepted value is shown. 49
Figure 4.16 The average accepted objective function value and the average best objective function value of 24 runs with 250 iteration and 50 moves with reheating. The standard deviation of the average accepted value is shown. 49
Figure 4.17 The average temperature and acceptance rate for the processes with and without reheating. 49
Figure 4.18 Average objective function value as a function of the design size. Designs with between 300 to 610 balls have, on average, better performance than the smaller and bigger designs. 50
Figure 4.19 Designs showing hopping locomotion. Design A uses the impulse of its arms to lift its base of the ground. Design B hops by lifting off and rotating before landing. The locomotion of design C is comparable to design A. Design D pushes off with its only leg and balances using a tail-like structure. 50

Figure 4.20 Walking designs. Although the propulsion of design A is very similar to that of hopping designs A and C, here, the legs are lifted off the ground alternating, resulting in steps. Designs B, C and D show legged locomotion where the leg is lifted of the floor just enough to reduce the friction. 51
Figure 4.21 A crawling and a rolling design. Design A crawls and has minimal lift-off of the ground. Design B rolls. 51
Figure 4.22 Average improvement of the objective function value for the different rules. All rules have, on average, a negative effect on the objective function value. A small percentage of rule applications improves the designs. The median is shown as a dot, the standard deviation as a bar. 53
Figure 4.23 The design entropy versus the average objective function value. The design entropy is defined as the average of the local diversity within a design. For every ball in the design, the local diversity is defined by the fraction of identical neighboring balls. 53
Figure 5.1 An overview of CDS with predefined actuation, as presented in Chapter 4 and CDS with actuation optimization. 56
Figure 5.2 a) The building blocks of the spatial grammar. Spheres and cubes are single elements. Cylinders are part of bending sub-assemblies. Every building block has 6 locations to connect to other building blocks. b) The spatial grammar rules used for the design generation. 58
Figure 5.3 Six activation patterns. The bottom three are phase-shifted duplicates of the top three and are used to generate an alternating gait in symmetric designs by assigning phase-shifted patterns to the mirrored actuators. 58
Figure 5.4 These designs, resulting from the generation with fixed actuation, are used in all control optimization experiments of Section 5.2. Design D has eight actuators, the other designs have two actuators. The objective function value f after 20 seconds of simulation with fixed actuation patterns are given as well. Orange balls and cubes are unactuated, stiff material. The rows of cylinders forming actuators are shown in different colors. Actuators that are symmetric counterparts are shown in the same color. 58
Figure 5.5 One and a half cycles of an order 15 Bezier curve. Every set of 3 Bezier Curve Points (CP) is aggregated and represented by a single variable for optimization. The last set is always equal to the first to ensure smooth cyclic behavior. By using an order 15 curve, the curves follow the curve points closer and the first and second derivatives at start and end are matched. 61

Figure 5.6 Average accepted objective function value, and the range, versus moves for PAC optimization. A PAC is optimized for Design A, Design B and Design D. For each design, the average accepted objective function value and the range of 22 runs is shown. 62
Figure 5.7 A schematic of actor-critic RL. 63
Figure 5.8 The four components of the input state; the actuator angle, the difference between the current and previous angle, floor-contact points and three rotations for the center building block. 63
Figure 5.9 Average objective function value plotted against state transitions for 9 SAC runs for Design A, Design B and Design C. For each designs, the objective function value achieved with fixed actuation is plotted as red dashed line. 63
Figure 5.10 Target angles for the 2 actuators of Design A given by the policy learned with SAC. As can be seen, the actions of the 2 actuators only differ slightly and the control is cyclic. 65
Figure 5.11 Schematic of CDS with an inner loop for actuation optimization. . . 66
Figure 5.12 Pseudo-code of the iterative design method. 68
Figure 5.13 The average and range of the best objective function value versus the iterations is plotted to compare the design of morphology only, morphology and actuation combined and the design of morphology with PACs as fixed actuation. The design process of morphology only is run 30 times and has 100 iterations and 25 moves per run. The design process with PAC optimization is run 30 times and each run has a maximum of 100 outer loop iterations with each 3 moves and a maximum of 300 inner loop iterations with each 5 moves. The process with actuation optimization is terminated when the convergence criteria are met. 69
Figure 5.14 Three results from the design of morphology and control are shown. The top design has six actuators, where the symmetric actuators are actuated in phase. It uses two sets of actuators to produce a hopping motion and an extra set to keep its balance. The middle design rolls. The bottom design has four symmetric actuators in counter-phase. One pair of actuators releases the pressure on one side of the body, the other pair pushes off. For each design, the Objective Function Value (OFV) is given. 70
Figure 5.15 Seven designs moving from left to right. The Objective Function Value (OFV) is given on the left. Designs a and e jump, designs b, c and g walk and designs d and f roll. 71

Figure 5.16 a) The hand-designed physical soft robot presented by Shepherd et al. and the same design generated by hand-picking grammar rules. b) The alternating gait (top) and the undulating gait (bottom) of the physical robot from [9] and the generated equivalent. 73
Figure 6.1 A pneumatic actuator realized through additive manufacturing. . . . 79
Figure 6.2 A step in the printing process on the From 2, image from [88]. A laser mask projected on the bottom of the tray cures a single layer of the design. To detach the newly cured layer from the tray, the tray slowly slides to the right before the build-plate is lifted from the resin. 79
Figure 6.3 An early iteration pneumatic actuator is shown in (a), the same actuator is shown in the top view (b) and the bottom view (c). The narrow parts of the actuator are partly filled with (cured) residue resin. Figures (d) and (e) show the cross-section as designed and as printed. The cross-section is filled with support columns that make cleaning near impossible. 80
Figure 6.4 Pneumatic prototype printed in one go on the Formlabs SLA machine. 80
Figure 6.5 The system of valves to control the pneumatic actuators. Picture from [90]. 81
Figure 6.6 Tendon actuators. An actuator wired on one side in a) un-actuated position and b) actuated postion. c) A close-up showing the inserts. Pictures d) and e) show an actuator wired with an elastic thread on one side and a non-elastic thread (green) on the other. 81
Figure 6.7 a) The setup of one motor as used for the actuation of the tendon-driven prototypes. To achieve enough thread displacement, the servomotor is mounted by a wheel. b) A printed joint to connect the pipes and guide the thread through the middle of the joint. 82
Figure 6.8 Four pneumatically actuated designs, simulated and prototyped. . . 86
Figure 6.9 Three tendon actuated designs, simulated and prototyped. 87
Figure 6.10 The speed in robot lengths of the different prototypes compared to several other soft robots. The referenced robots can be found in [21], [10], [92], [9], [22] and [39]. 88
Figure 7.1 Overview of the CDS method presented in this thesis. 93
Figure 7.2 Three designs with the same objective function value yet very diverse morphologies and gaits. All three designs have an objective function value of 76. 97
Figure 7.3 Four designs resulting from the CDS method presented in this thesis that show locomotion for which a biological analogy is hard to find. For each design, the gait is described briefly and the objective function value (OFV) is given. 101

Figure A.1 Using soft body dynamics from the Bullet Physics Library to simulate (a) inflating designs through surface forces and (b) bending designs through distance constraint manipulation. The colors in both pictures signify material stiffness and actuation. 108
Figure A.2 Two line segments with different resolution illustrate the difference in error correction propagation. Each segment consists of 2 or 3 vertices, connected by springs (shown as lines). Here t is the time-step, or constraint solver iteration step. The two red vertices that penetrate the wall are in violation of the collision constraints. Their collision constraint violation is solved in the first constraint solver iteration. Their movement is only noticed by neighboring constraints during the consecutive iterations. The low resolution segment updates the complete mesh in less iterations. Therefore, the low resolution segment will behave stiffer for the same number of constraint solver iterations. 109
Figure A.3 Constructing the ray to be checked for Ray-Triangle Intersection. a) Determining the relative position of point v with respect to the face p_1, p_2, p_3 at time-step t_0. The relative position is denoted by r_0. b) Defining the ray r_0 - v to check against face p_1, p_2, p_3 at t_1. 111
Figure A.4 An example of triangle-triangle collision without point-triangle collision. 111
Figure A.5 Walking robot designs simulated using position-based dynamics. . . 112
Figure B.1 Mean best objective function value versus state transitions of 21 runs of DQN for Design A. The objective function value of the design with fixed actuation is shown in red dashed line. 115

LIST OF TABLES

Table 4.1 Performance measures for different optimization processes. It shows that long runs have better results on average. However, the difference between the maximum values is small. 37
Table 4.2 The occurrence of the different classes, in a set of 105 manually classified designs, and their performance measures. 37
Table 4.3 Occurrence of rules in the history of final designs. Note: when designs exceed the maximum size, the deletion rule is selected. . . . 42
Table 4.4 Results for 3 optimization processes, each consisting of 24 runs. The first column describes the parameters used for the sets of 24 runs. In the subsequent three columns, the average best objective function value, the minimum objective function value and the maximum objective function value are reported. The last column shows the standard deviation of the best objective function values. 48
Table 4.5 Occurrence of rules in the history of final designs. This data is generated by recording accepted rule applications only. Note that when designs exceed the maximum size, the undo-rule is chosen. . . 53
Table 5.1 The PAC optimization results compared to the fixed actuation result for three designs. 61
Table 5.2 The SAC results compared to the fixed actuation result for three designs. 64
Table 5.3 Comparing the performance of PAC optimization and SAC for finding the actuation of Design A. The relevant measures for computational cost differ between the methods but are related through simulation length and time step. The converted measure, related through the simulation time and step size, is given in parentheses. . 65
Table 5.4 Parameters used for the method given in Algorithm 5.12. 67
Table 6.1 The predicted performance and the performance of the prototypes shown in Figure 6.8. 85
Table 6.2 The predicted performance and the performance of the prototypes shown in Figure 6.9. 85
Table 6.3 The predicted performance and the performance of the prototypes shown in Figure 6.8. 88
Table 6.4 The novelty scores of the prototyped designs. The first row reports the average novelty of all the designs in the comparison set. The novelty scores of the individual benchmark designs can be found in Appendix C. 89

Table 7.1 This design method proposed in the work in comparison to related work on the automated design of soft robots. 100
Table B.1 The DQN results compared to the fixed actuation result for Design A. 114
Table C.1 The collection of soft robots used for the novelty score calculation in Chapter 6.3.1. 118

PREFACE

Parts of this dissertation are adapted from the publications listed on page 129 with the permission of the publishers. If a whole chapter is based on a publication, it is indicated at the beginning of the chapter, which publication the content is adapted from.

1

INTRODUCTION

In many robotic applications, a compliant robot is preferred over a completely stiff robot, for example to increase safety when a robot has direct interaction with humans [1] or when a robot grips an object with an unknown shape [2] [3]. Where a stiff robot will oppose any external forces, a compliant robot yields to a certain extent. In position-controlled robots, compliance can compensate for inaccuracies of the robotic components and the perception of the environment. Additionally, compliant components can absorb energy from collisions and even store energy. Compliance can be achieved actively or passively. Active compliant robots have a controller that provides the compliant behavior, for example by measuring contact forces, whereas passive compliant robots make use of elastic elements that are compliant. Where active compliant system rely on sensors and computations, the passive compliant ellements in passive systems make these computations redundant and reduce the number of control variables [4]. Therefore, passive compliance is sometimes referred to as mechanical computation [5]. As shown by [6], passive compliant grippers are versatile and a good alternative to active compliant grippers relying on force feedback.

Within passively compliant robots, a distinction can be made between robots using compliant elements, e.g springs and dampers, and robots that have bodies made out of compliant material, e.g. silicone rubbers. Although the term 'soft robot' is sometimes used for all compliant robots [7], here it is solely used robots that achieve compliant behavior through the use of compliant materials for their bodies; see also [8]. There are several advantages of soft robots. For example, when gripping arbitrarily shaped objects, like fruit, the continuous deformation of the completely soft gripper arms allows for a better distributed gripping force. Their flexibility can allow soft robots to reach places that are challenging for rigid robots; see, for example, the quadruped from Shepherd et al. [9] crawling through a narrow opening. Currently this flexibility is not fully exploited, but it is possible that the limitations on the flexibility of future soft robots will only be due to the necessary electronics, not unlike how an octopus fits through any opening large enough to fit its beak. The absence of rigid joints in soft robots is an advantage since joints are easily damaged by liquids and dirt. Additionally, most actuation methods used in soft robotics allow for designs that can withstand extreme conditions when not actuated. An example of this is the pneumatically actuated quadruped from Tolley et al. [10]. The soft robot is shown to survive a car driving over its deflated legs. The most common actuation methods used in soft robotics are pneumatics, tendon-driven actuation, fluidic elastomer actuation and shape memory alloy actuation [11].

Soft robots have found their way into industrial applications as robotic grippers, for example in horticulture as illustrated by the produce gripper shown in Figure 1.1. For soft robotic grippers to interact in a human environment and be able to grip objects that were designed to be gripped by human hands, versatile and dexterous grippers are required. Gripper designs aiming for versatility and dexterity can take the form of hands [12], trunks [13] and starfish-grippers [14].

There is also a growing interest in soft robots for locomotion. Locomotion is an essential component of common robotic tasks such as search and rescue, the exploration of remote areas and pick and place. This thesis will concentrate on the task of locomotion.

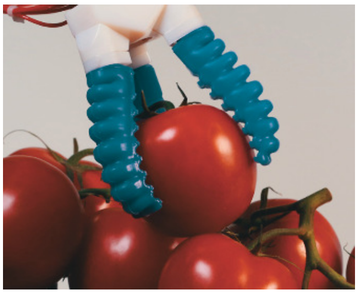


FIGURE 1.1: An example of a soft gripper for food handling, from [15].

A number of soft robotic designs have been developed in the last 15 years. However, the large potential of soft robots for locomotion is not yet exploited. Figure 1.2 shows several examples of soft robots and an extensive overview can be found in [8].

One of the most challenging aspects of soft robot design is fabrication. The main fabrication methods are silicone casting and, more recently, additive manufacturing. Casting silicone soft robots sets many restrictions on the design in order to be able to eject the parts from the molds and assembling silicone parts introduces irregularities and is time-consuming work. On the other hand, many of the compliant materials available for additive manufacturing either degrade rapidly or are too stiff for most soft robot designs. Recent advances in printing of of-the-shelf silicone, with the process described in [16], brings the fabrication of free-form soft robots a step closer. And even then, these fabrication processes will likely limit the designs, e.g. the size of designs is limited to the print plate size, the design has to be strong enough to support every printed next layer, whether or not a support structure is added, and the design has to allow for cleaning of residue printing material. The last point is especially critical for hollow designs.

Assuming these fabrication issues can be solved, the next challenge is the design process of soft robots. Designing soft robots by hand and using a trial-and-error process is time-consuming, as is confirmed by literature [17]. The dynamics of the elastic bodies are hard to predict and the design freedom is vast. This can make the first phase of a design process, namely settling on a conceptual design, confounding. A majority of the current soft robot designs is therefore bio-inspired. The locomotion of an animal is taken as base for the conceptual design, as this selection of soft robots shows: octopus [18], caterpillar [19], mesh-worm [20] and starfish [21]. Although this approach leads to many interesting locomotive soft robots, concept selection is a crucial part of design and deserves careful consideration. Animals are shaped the way they are as a consequence of many factors, e.g. their energy source, their temperature regulation in conjunction with the heat generated by their cells, the environment they live in and their co-inhabitants. None of these constraints hold for soft robots and the design space of animals is not equivalent to the design space of soft robots.

This poses the question as to whether there is a way to devise conceptual designs that takes into account the specific opportunities and limitations of soft robots. Additionally, one can ask how the selection of a concept can be justified, i.e. how can different concepts, bio-inspired or other, be compared so that the designer can make an informed choice? In an attempt to offer a solution to these two questions, this thesis proposes a Computational Design Synthesis (CDS) approach for the conceptual design of soft robots for locomotion.

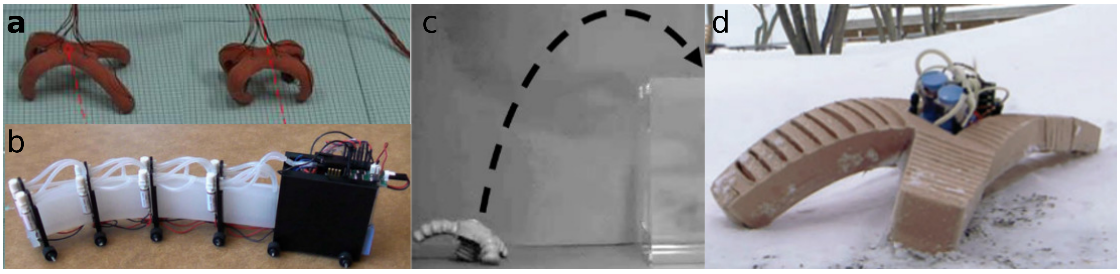


Figure 1.2: Several examples of soft robots for locomotion. a) A tetrapod actuated by shape memory alloys [21], b) a snake-bot using fluidic elastomer actuators [22], c) a pneumatic jumping robot [23] and pneumatic tetrapod for walking [10].

CDS is an iterative process that generates, evaluates and optimizes design candidates. It can be used to automate time-consuming design tasks. As opposed to parametric optimization, a synthesis generates new concepts and can therefore help a human designer overcome fixating on known solutions and explore a design space more efficiently [24]. The computational design of soft robots is still relatively unexplored. Related work mostly uses the perspective of artificial life [25], [26], [27], [28], a field of research examining natural life and evolution through simulation. That type of research does not aim to support soft robotic engineering and consequently, there is little emphasis on the feasibility and practicability of the resulting designs. To design soft robots within an engineering context, three additional requirements are important.

First, the generated designs must be feasible, i.e. it must comply with fabrication constraints, size constraints and limitations on the available energy input.

Second, the design process itself must be transparent. The conceptual designs from an automated design process have to be interpreted, detailed and fabricated by a human designer. As design is inevitably an iterative process, new insights resulting from detail design and fabrication can be applied to the concept generation. Therefore, the design generation method must give the designer some measure of control over the design process. In related work, design generation with a type of neural network [27] or L-system [28] can be found. These types of generation methods result in black box design generation. It is not intuitive for a designer to modify these methods such that they include the observations from the detail design and fabrication phase.

Third, it is important to strike a balance between the novelty of a design and the chance the design will be accepted by the designer. Currently, computationally generated designs are very different from designed by humans. To illustrate this, Figure 1.3 shows computationally synthesized designs to compare to the hand-designed robots from Figure 1.2. What stands out is that hand-designed soft robots often have legs whereas computationally designed soft robots have either no extremities at all or very short ones. On the one hand, generating novel designs and overcoming the human design-bias are key functions of CDS, on the other hand, a designer might prefer a design that can be fabricated with known fabrication methods or a design that is familiar enough so the designer can modify and tune it.

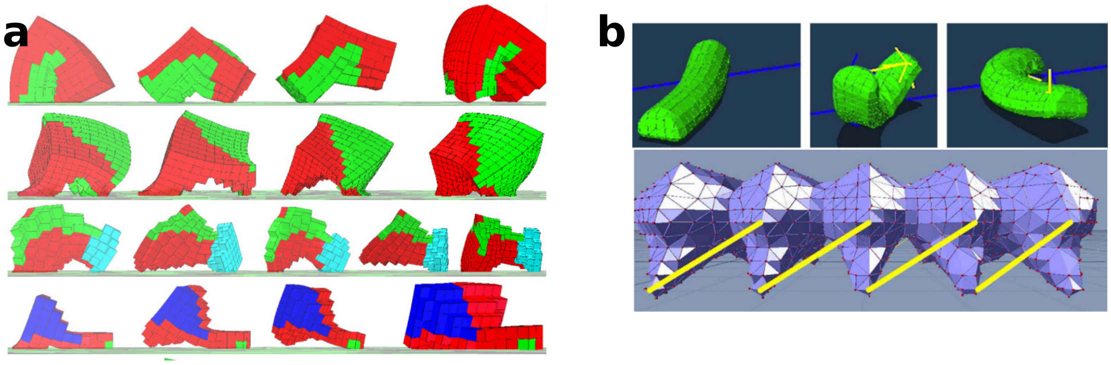


Figure 1.3: Soft robot designs resulting from automated design processes. a) Several of the designs resulting from the computational design synthesis of soft robots from [27], b) the two results from the computational design sysnthesis reported by [28]. Comparing these designs to the designs in Figure 1.2, the lack of extremities stands out.

Next to a morphology, a controller is required to control the actuation of the soft robot. Additional to the vast design solution space formed by the morphology design problem, the possible ways to actuate the morphologies multiplies the number of possible resulting designs vastly. A common approach is to limit the possible actuation modes and concentrate on finding a morphology design that succeeds with a given actuation [25], [26], [27], [28]. However, it is difficult to deduce what the implications of this limitation on the design space are, i.e. which designs are excluded through the choice of actuation, even though they might perform well with a different actuation? Since designing morphology and control simultaneously is a much more comprehensive approach, it is introduced as a fourth requirement for the design generation approach for the design of soft robots.

1.1 SCOPE OF THE THESIS

This thesis investigates the CDS of soft robots for locomotion, specifically focusing on the generation of concepts for soft robots in an engineering context.

The objective of this thesis is: **to develop a computational design synthesis method for the conceptual design of the combined morphology and actuation of soft robots for locomotion**.

This objective is split up in three stages, 1) the first stage focuses on the design of morphologies only, 2) the second stage investigates the co-design of morphology and control and 3) the third stage concludes the work by demonstrating the detail design and prototyping process of the designs resulting from the approach.

To achieve the objective, the following **research questions** need to be answered.

1. The design of morphologies only:
   - 1.1 How can a morphology generation process be created that can be controlled and guided, while still maintaining enough design freedom?
   - 1.2 Which simulation model best supports the evaluation of soft robots concepts for locomotion within CDS?
   - 1.3 What are the characteristics of the solution space and which optimization method is suitable to carry out the optimization within CDS?
2. The co-design of morphology and actuation:
   - 2.1 How can an actuation pattern be optimized for an arbitrary design within a limited time frame?
   - 2.2 How can the search for an actuation pattern for every design candidate be integrated in the CDS method?
3. The detail design and prototyping of the designs:
   - 3.1 How can the design concepts be detailed and prototyped?
   - 3.2 How can insights from the process of prototyping be integrated directly in the CDS method?

1.2 EXPECTED CONTRIBUTIONS

The expected contributions of this thesis are:

- A generative design method for the generation of soft robots morphologies for locomotion that takes into account design requirements and constraints.
- A model for the simulation of soft robot that is stable for generated designs and is computationally efficient enough for the integration into an iterative design method.
- A method for the optimization of the actuation pattern of designs for locomotion that strikes a balance between computational cost and performance and does not need design-candidate specific tuning of optimization parameters.
- The integration of this actuation optimization method in the CDS method of soft robot concepts is demonstrated.

- The detailing and prototyping of generated designs to show the capability of the approach to generate concepts within an engineering context.
- Prototyped that perform comparably well to state-of-the-art soft robots and are highly novel.

1.3 DESIGN TASK FORMULATION

The generative method presented in this thesis generates conceptual designs for the problem of soft robot locomotion. Within a finite time, designs are to move as far as possible, i.e. they are optimized for speed. Only the effective distance traveled in the horizontal plane, in any direction, is considered. This effective distance is defined as the difference between the starting point and the end point, as opposed to integrating the path of the robot. To prevent counting rotation as locomotion, the effective distance traveled by the point of the robot that traveled the least is taken as the distance measure. This objective is illustrated in Figure 1.4 where f_1 is defined by the distance traversed by the point that is closest to its original location. Since the evaluation time is limited, designs that convert their potential energy into kinematic energy, i.e. designs that fall, have a change to receive a high performance value. Therefore, a second objective, f_2, is defined as the average lost potential energy of the elements of the design and this objective is to be minimized, see Figure 1.4.

The objectives described here are formally expressed in an objective value function given in Chapters 4 and 5.

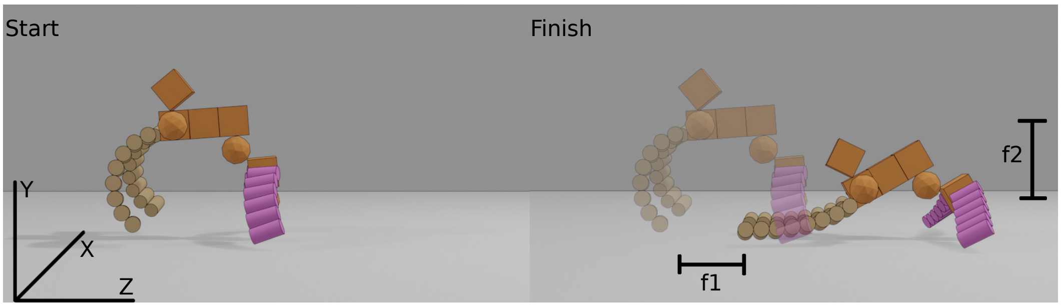


FIGURE 1.4: A robotic design is used to illustrate the two objectives of the optimization. The first objective, f_1, is the shortest distance traversed by any point of the design in the horizontal plane and is to be maximized. The second objective, f_2, is the average lost potential energy of the elements of the design and this objective is to be minimized.

1.4 THE COMPUTATIONAL DESIGN SYNTHESIS METHODOLOGY

A CDS consists of the three design stages that are iterated until the design requirements are fulfilled [29]; generate, evaluate and guide. Here, this methodology is implemented for the design of soft robots for locomotion and the three stages look as follows.

DESIGN GENERATION A design generation method creates new designs, either by adapting a previous design or by creating an original design. The generation method operates on a representation of the design. How designs are represented has a large influence on the design freedom and on what designs are more likely to be generated, i.e together with the objective value function, the selected representation outlines the solution space.

EVALUATION Designs are evaluated and assigned an objective function value. In the work presented here, evaluation is realized by simulating the designs for a fixed duration and expressing the performance through an objective value function that measures the distance traveled by the design. To evaluate the performance of a robotic design in simulation, the robot requires a controller. Here, this actuation pattern is a fixed predefined actuation pattern in Chapter 4 and a actuation pattern tailored to the simulated morphology in Chapters 5 and 6.

OPTIMIZATION An optimization method guides the search for the optimal design in the solution space. Here, the heuristic search method simulated annealing is selected.

The approach presented in Chapter 4 consists of the three stages described above. In Chapter 5, the approach is extended with an extra stage for finding an actuation pattern for every morphological design.

ACTUATION PATTERN OPTIMIZATION Instead of actuating a design according to a predefined actuation pattern, an actuation pattern is optimized for each individual design.

Figure 1.5 shows an overview of the two design processes, one without design specific control and one with design specific control.

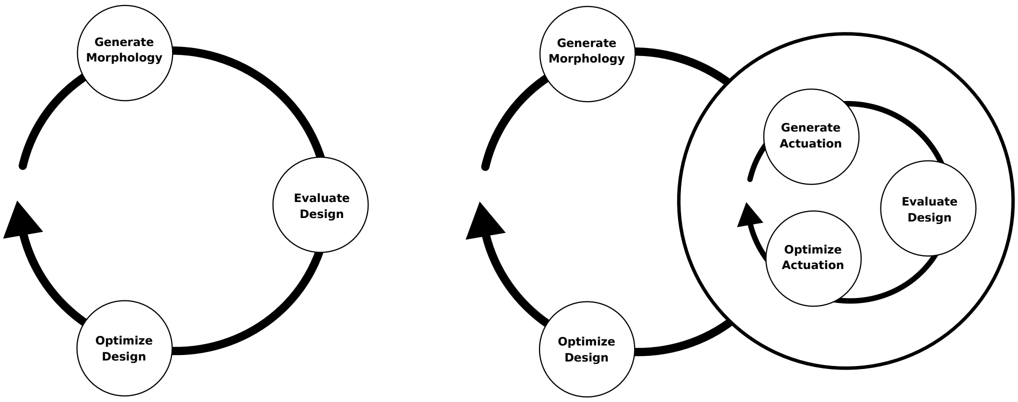


FIGURE 1.5: Overview of the computational design synthesis approach with and without design specific control.

1.5 DISSERTATION OVERVIEW

The dissertation is structured as follows. Chapter 2 presents the relevant background and discusses the selection of the different methods used in the CDS. To determine a simulation method suitable as part of the evaluation of the designs in the generative design method, Chapter 3 presents different models and discusses their performance. A generative design method designing morphologies, actuated by a non-varying actuation pattern, is presented in Chapter 4. Chapter 5 extends this work by integrating actuation pattern optimization for every individual design to the generative design method. Additionally, an alternative simulation model is used, shifting the focus from approximating soft material behavior to approximating existing actuator behavior. Chapter 6, presents the detail design and prototyping of the computationally generated designs. The thesis is concluded and future work suggested in Chapter 8. A thesis roadmap is provided in Figure 1.6 and an overview of the iterative design process, including the chapters per design stage, is presented in Figure 1.7.

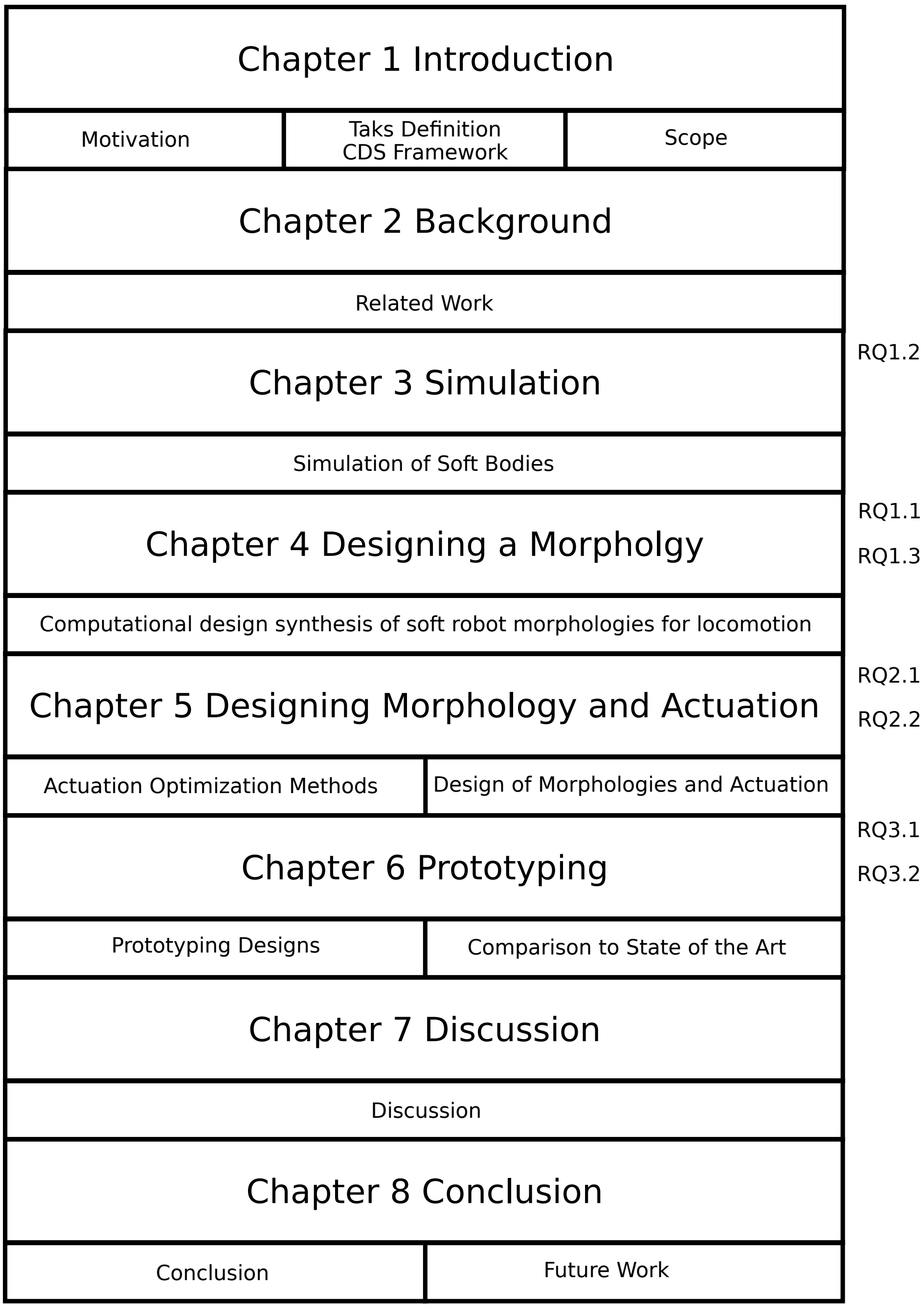

Chapter 1 Introduction
Motivation
Taks Definition
CDS Framework
Scope
Chapter 2 Background
Related Work
Chapter 3 Simulation
RQ1.2
Simulation of Soft Bodies
Chapter 4 Designing a Morpholgy
RQ1.1
RQ1.3
Computational design synthesis of soft robot morphologies for locomotion
Chapter 5 Designing Morphology and Actuation
RQ2.1
RQ2.2
Actuation Optimization Methods
Design of Morphologies and Actuation
Chapter 6 Prototyping
RQ3.1
RQ3.2
Prototyping Designs
Comparison to State of the Art
Chapter 7 Discussion
Discussion
Chapter 8 Conclusion
Conclusion
Future Work

FIGURE 1.6: Thesis roadmap, including where the research questions are answered.

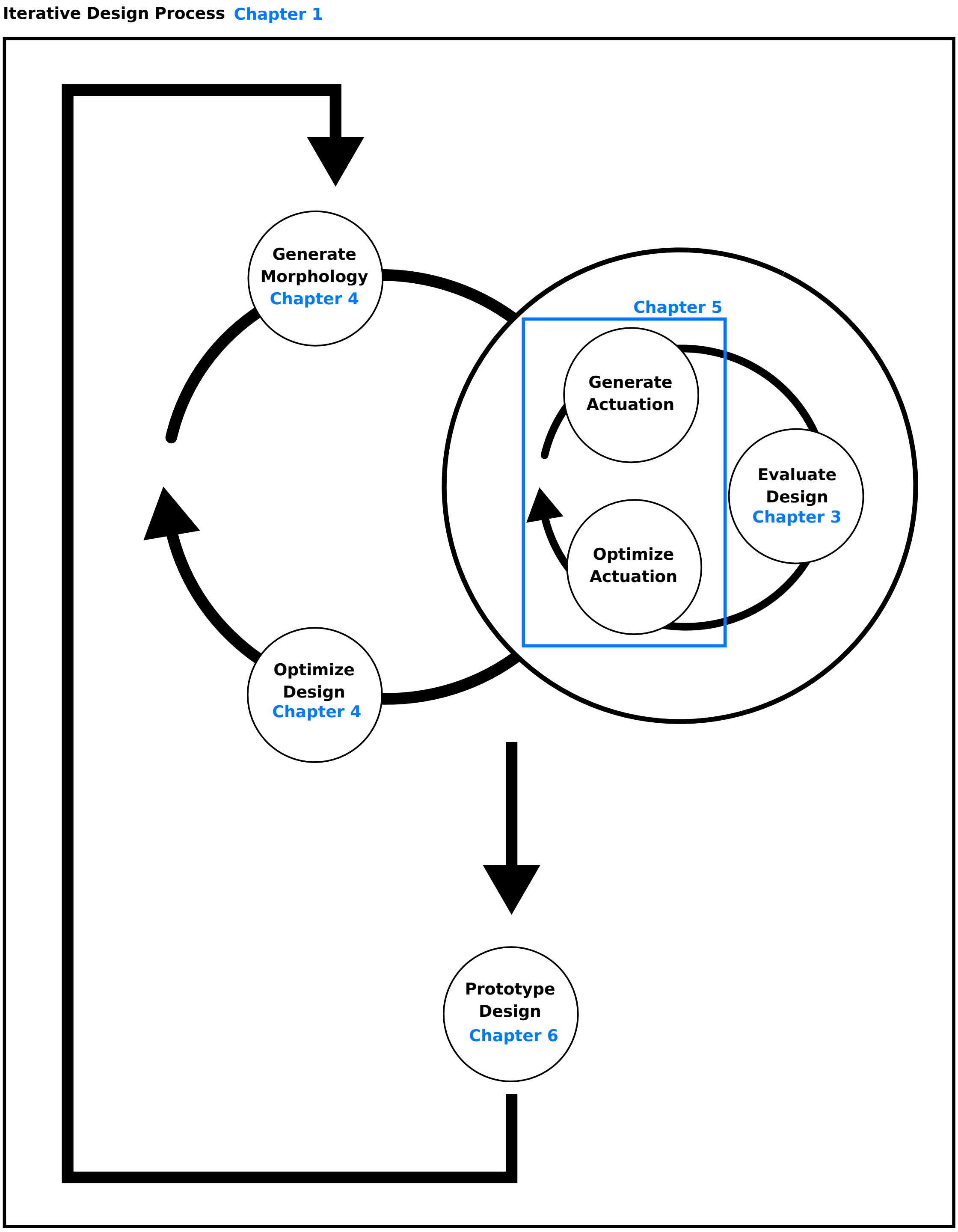

Iterative Design Process Chapter 1
Generate Morphology Chapter 4
Chapter 5
Generate Actuation
Evaluate Design Chapter 3
Optimize Actuation
Optimize Design Chapter 4
Prototype Design Chapter 6

FIGURE 1.7: An overview of the iterative design process, including the chapters further details on the design stages can be found.

2

BACKGROUND

This chapter starts by presenting the related work on soft robots and design automation. It follows by discussing the literature on the automation of actuation within a CDS and concludes with the implications of the related work to the research conducted in this thesis.

2.1 RELATED WORK ON SOFT ROBOTS

This section discusses the related work on the design, fabrication and control of soft robots.

2.1.1 *Soft Robot Design*

By far most of the soft robot designs found in literature are presented as bio-inspired design. Many researchers find inspiration in worms [20] and caterpillars [19] [30] as locomotive designs. There are octopus-inspired robotic arms [31] [32] and a robotic octopus without a function [18]. A fast land-based robot for locomotion is inspired by snakes [22] and several water-based locomotion robots by fish [33] [34]. In the case of the four-legged robot by [9], biology as a whole is mentioned as inspiration. For some designs, like the jumping combustion robot presented by Bartlett et al. [35], the origin of the concept is unknown. An overview of bio-inspired soft robot design is also found in [11] and [36].

Although there are many reasons to take biology as an example, robots also enable solutions that are impossible with biological systems. Therefore, the search for a robotic design should not be restricted to biology. Additionally, the reviewed literature does not provide a rationale for the selected bio-inspiration. There is no comparison between all animals that can perform the design task, nor between the different motions these animals exhibit. Why take a worm over a caterpillar as inspiration for a locomotion robot?

A challenge in designing soft robots is that the field is relatively new and, compared to the field of rigid robots, small. Consequently, there are no off-the-shelf components yet and every soft robot designer develops a new, often difficult to replicate, prototyping method. Additionally, the simulation of soft robot designs is challenging, as is reviewed in Section 2.2.2 and further explored in Chapter 3. Due to these two challenges, the design of soft robots is a trial-and-error process that is time consuming. These challenges and the lack of an alternative approach might be part of the motivation to use biological analogies as conceptual design.

A handful of attempts to automate the design of soft robots is made and these efforts are described in detail in Section 2.2 of this chapter.

2.1.2 *Soft Robot Fabrication*

Soft robots are fabricated with a variety of soft materials. An overview of material selection and actuation specifically for bio-inspired robots can be found in [36]. A fabrication method growing in popularity is additive manufacturing. Here, the selection of the elastic material is combined with the selection of the printer, since most printers are restricted to use proprietary material developed specifically for that printing process. Examples of 3D-printed soft robots are ample. Additively manufactured caterpillar-like robots are found with shape memory alloys [37] [30] and tendon-actuation [38]. A two-legged pneumatically actuated crawler is printed by DuPasquier et al. [39] and a combustion actuated jumping robot is presented by Bartlett et al. [35]. Although additive manufacturing promises an almost unlimited design freedom, the examples illustrate that most designs have conventional morphologies and do not use this freedom yet. An exception to that is the voxel-based design presented by Hiller and Lipson [40]. However, their robot, consisting of a collection of open-cell and closed-cell foam rubbers, is activated by lowering the pressure in a pressure chamber.

Until additive manufacturing machines that use off-the-shelf materials are more readily available, designing robots with known and stable material properties relies mostly on molding with silicone rubbers. This process is demonstrated for, among others, a four-legged robot [10], a caterpillar [19], a rolling robot [41], a jumping robot [23] and a swimming robot [42]. Molding, however, requires robot parts to be ejected from the molds, thus restricting the design. One way to increase design freedom using molding is designing robots consisting of inflatable cubes [43]. Hollow silicone cubes are fabricated using a 1-axis rotational drip molding machine. These cubes can be inflated individually and function as pneumatic actuators. Although there is much manual assembly involved, the molding process is repeatable and predictable, and a variety of different designs can be fabricated.

2.1.3 *Soft Robot Control*

Due to the highly elastic material in soft robots and the presence of (self-) collision that is challenging to model, the design of soft robot controllers often relies on simulation-based optimization. Ishige et al. [44] control a caterpillar-like soft robot with a Central Pattern Generator (CPG) and optimize the controller parameters using reinforcement learning. Rieffel et al. [45] use a spiking neural network to control the shape memory alloy muscles of a caterpillar-like robot. The weights of this network are optimized with a genetic algorithm. Both examples show feedback controllers with a small set of parameters. By knowing the morphology and the desired behavior of the robot, assumptions can be made to reduce the control problem. However, when morphology and desired behavior are unknown, these methods are infeasible. Spielberg et al. [46] present a method that simultaneously learns a latent state representation and a control policy. An auto-encoder is trained to reduce the state of a simulated soft robot and a neural network is trained to select the best action given a current state.

2.2 RELATED WORK ON DESIGN AUTOMATION

The automated design of virtual robots was pioneered by Sims [47], although he presents his work as part of the field of artificial life. A rule-based system operating on directed graphs is used to generate robot morphologies and neural control circuitry. The graph representation of a design consists of morphological nodes with nested graphs of neural nodes. The morphologies described by such a graph consist of rigid blocks connected by dimensionless joints. Each block has a dedicated neural circuit, defined by the nested graph of neural nodes, that is connected to a central neural circuit. The nodal activation functions and connecting weights of the neural networks are randomly set during generation. The behavior of the design emerges from the combination of the nested network components. Although a controller follows from this process, the parameters of the neural networks are modified concurrently with the parameters of the morphology. Therefore, there is no control optimization for every morphology and potentially successful morphologies fail due to an unfitting controller. The designs are evaluated in simulation and scored based on their capacity to walk, jump or swim. A genetic algorithm is used to optimize the designs.

Similarly, Lipson and Pollack [48] use small neural networks embedded in the building blocks of the robot morphology to control evolved truss robots. Lund et al. [49] evolve rigid robot hardware with a genetic algorithm and simultaneously evolve the control using genetic programming.

Cheney et al. [27] argue that composing robots of rigid elements is too restrictive to allow for the emergence of more complex robots. Taking inspiration from developmental biology, they present a computational design method for soft robots consisting of voxels of material with different stiffness. Four predefined actuation patterns are used to expand and contract the voxels of soft material. The actuation patterns stay unmodified, it is the assignment of the four patterns to the different voxels that is subjected to change. The resulting designs showcase a variety of morphologies and behavior that differs strongly from earlier work. The authors present the resulting designs as a way to fill artificial worlds with creatures. As such, the designs do not necessarily inspire or support physical robotic design.

A similar observation is made about the soft robots designed by Rieffel et al. [28]. The virtual designs are actuated by dimensionless, linear actuators they refer to as muscles. The morphologies are generated by an L-system like generation method and the actuation of the muscles is defined by an initial muscle firing pattern. The designs are evaluated in simulation. With a genetic algorithm, the morphology is optimized for a fixed number of generations. Subsequently, the morphology is kept constant and the firing pattern is subjected to optimization. Iterating these two phases, a correspondence between morphology and control is found. This generative method is shown the evolve novel designs, however, the generation method is difficult to manipulate and the dimensionless muscles leave a large task for the detail design phase. Thus, this approach is not aimed at supporting engineering design.

The automated design of a virtual sailboat is shown by Stump et al. [50]. A grammar for sailboat components is proposed and a character-recurrent neural network is trained to generate strings of characters that represent successful designs. The sailboats are evaluated in a the Unity Game Engine. The control policy to control the rotation of each sail is found through reinforcement learning. Since reinforcement learning is applied to each design candidate, designs are simultaneously optimized for form and behavior.

2.2.1 *Design Generation Methods*

This section presents the different design representation and generation methods for the design of soft robots found in literature. A design representation is in part responsible for the shape of the solution space of the problem. Any representation excludes certain designs and makes other designs more likely to be generated. This effect can be used to increase the generation probability of desired designs and exclude infeasible designs from the solution space. The generation method operates on the representation of the designs, thus, the selection of the generation method follows from the selected representation method.

The first consideration when defining the representation of the robotic designs is the level of abstraction. A representation with a low abstraction level represents all features of a design as independent pieces of information. As an example of a low abstraction representation, one can think of representing an animal as a list of body cells. This so called direct encoding is shown to create a combinatorial problem that is too large to solve with the recourses known to us [51], [27]. An example of a more abstract representation of an animal is a list of body parts. With a well chosen representation, nonsensical designs can be excluded from the solution space while there is still space for novel designs.

The second consideration is how modification to an indirectly encoded design correspond to changes in the designs morphology. A solution space is more smooth when a small change in the representation corresponds with a small change in the morphology. Additionally, some representations allow for local changes, where others can only modify the whole design.

Hiller et al. [52] generate morphologies consisting of soft, volumetric expanding materials. The designs are represented by a list of Gaussian points in 3D space. These points define the density of different material. The coordinates and strength of the Gaussian points are subjected to mutation with a genetic algorithm. Depending on the number of points used, this indirect representation allows for local changes. Interestingly, the resulting designs often have a square silhouette, presumably due the boundary of the design space. They also discuss the possibility to encode designs through a Compositional Pattern-Producing Network (CPPN) and a Discrete Cosine Transform (DCT). Comparing the three design representations, they conclude that a DCT and CPPN are inferior since mutations always have a global effect on the morphology.

In later work [27], a CPPN representation is used to encode similar voxel-based morphologies. Here, the CPPNs are evolved with the NEAT algorithm [53]. The material of every voxel is defined by querying the CPPN. The coordinates of a voxel are the input to the network, the output determine the presence of material at that location and the properties of that material. If a modification to the nodal function of the network result in a global or a local change of the morphology depends on the size of the network and the network layer the modified node belongs to.

Rieffel et al. [28] generate designs using a system closely related to L-systems and provide little discussion on the effectiveness of the method. In contrast to the methods discussed above, the designs resulting from this method suggest modifications are only local and the method lacks a global modifier that can bring coherence in the design. Dimensionless linear actuators provide the actuation of the designs in simulation.

As mentioned in Chapter 1, the design representation and generation methods in previous work fall short when used in engineering design. The volumetric expanding voxels can be 3D-printed but providing the actuation by placing the design in a pressure chamber is impracticable and allows for one actuation pattern only. Furthermore, the generative method designing robots with dimensionless actuators neglects the physical properties of real actuators and can quickly result in infeasible designs. Additionally, it is difficult to incorporate design constraints directly into these methods and they are not easy to manipulate, e.g. what can one do to force a CPPN to generate designs with longer extremities?

2.2.1.1 *Spatial Grammars*

The generation method for soft robots presented in this thesis is a spatial grammar. A spatial grammar is a type of generative system that consists of a set of rules that govern the transformation of shapes. Four kinds of spatial grammars can be distinguished: string grammars, set grammars, graph grammars and shape grammars [54]. Where string, set and graph grammars operate on a representation of the design, shape grammars operate on the shapes directly. Here, the relevant concepts for a shape grammar are introduced since the grammar used in this work is closest to a shape grammar.

A shape grammar rule consists of a Left Hand Side (LHS) and a Right Hand Side (RHS). The LHS shows the shape that is eligible to be transformed by this rule, the RHS shows the shape that will replace the original shape after rule application. In the case of a parametric grammar, as is used in this work, the RHS is further defined by a set of parameters. An example of a spatial grammar rule and an application of that rule is illustrated by Figure 2.1. The grammar proposed in this thesis applies the rules sequentially, i.e. one rule application results in a single modification. A grammar has an initialization rule that creates an initial shape. Every consecutive design iteration, a rule is selected and applied once to the current design.

Stiny and Gips [55] first introduced the shape grammar formalism. Since then, shape grammars are successfully used for product design, [56] and [57], and architecture, [58]. In [59], an overview of implementations of spatial grammars, of which shape grammars

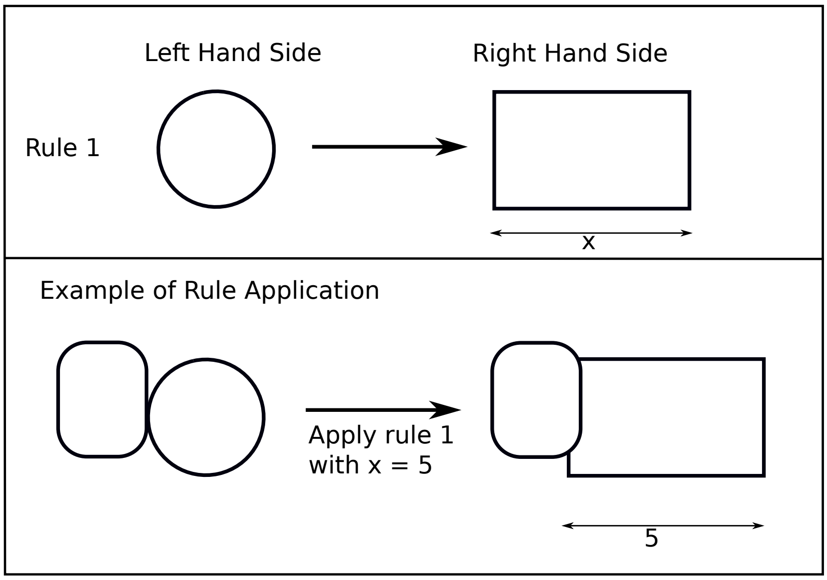


FIGURE 2.1: An illustrative example of a spatial grammar with one rule. The rule replaces a circle, the left hand side, with a rectangle, the right hand side. The width of the rectangle is dependent on the parameter X. An rule application example with $X = 5$ is shown at the bottom.

are a subdivision, can be found. Shape grammars combined with simulated annealing, called shape annealing, is shown to generate and optimize structural designs, see [60], [61] and [62].

An advantage of using a spatial grammar for design is that a part of the design requirements can encoded in the spatial grammar and be fulfilled during generation. An example is a limitation on the dimensions of a shape. Where it is unclear how to modify a CPPN to reduce the size of the corresponding design, the effect of a shape grammar rule on the design size is more predictable. Thus, the generation method can exclude undesired designs, rather than the computationally expensive evaluation stage.

The design representations used in this thesis are defined to be consistent with the model used to simulate the soft robots. Therefore, Chapter 3 introduces the used simulation models and, subsequently, the implementations of alternative representations and spatial grammars are presented in Chapters 4 and 5.

2.2.2 *Evaluation Methods*

The main method of evaluation in automated design of robots, as well as in evolutionary robotics, is simulation. A soft material simulator named VoxCad is presented by Hiller et al. [63] and is used to evaluate designs within a CDS. It is based on the finite elements method and is validated for a beam element. The designs generated by Rieffel et al. [28] are simulated with NVIDIA's PhysX. Of both simulation methods, it is unknown how accurately the simulations approximate physical soft robots, including friction and collision.

Other available simulators for soft material are Sofa [64], Bullet [65] and ChainQueen [66]. The latter is used for the simulation of soft robots in the process of optimizating their controllers [46].

The simulation methods mentioned here are only a portion of the simulation methods in existence at the time this research was conducted. However, to assess if a method is suitable as part of a CDS, it needs to be tested. Since implementing and testing is time consuming, the search for a suitable simulation method is not exhaustive. In Chapter 3, several methods are tested as part of a CDS and this shows that the tested methods require parameter tuning and fail to simulate specific cases, that are different for each method.

Although time consuming, iterative design using physical evaluation is shown to be feasible as well [67], [68]. Choosing to use physical evaluation means choosing to eliminate the reality gap at the cost of a smaller number of design iterations. The work presented here attempts to bridge the reality gap by tuning the simulation model rather than prototyping each design candidate.

2.2.3 *Optimization Methods*

The optimization problem presented here has a highly non-linear relation between the modification of the design and the resulting change in objective function value. Additionally, the variable space is discrete and unordered. These properties are the motivation to focus on meta-heuristic methods, the three most used methods being Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Simulated Annealing (SA).

When using a spatial grammar, a design is defined as a collection of shapes. The size of this collection can change over time. A PSO uses a dimension for each variable, however, does not allow for a change in the number of dimensions during optimization. Therefore, PSO is ruled out as optimization method for this particular design problem.

A GA has design modifiers defined in the form of mutation and crossover. When these modifiers are used to alter the design directly, they take the place of the spatial grammar. However, combining the use of mutation and crossover with the use of a spatial grammar is not straightforward. One could mutate and crossover sequences of rule applications, i.e. a sequence of assembly instructions, but that negates the benefit of 'live' designs. With live designs, designs are meant that manifest the features, for example dimensions, as a result of rule applications. An example of the use of these features is the decision to apply a spatial grammar rule that grows the design if the design currently is too small.

Simulated annealing is compatible with grammars and is selected to optimize the designs. Examples of processes using spatial grammars in combination with simulated annealing, also called shape annealing, have been used for the computational design synthesis of truss structures [61] and rollerblade wheels [62]. In [69], a method is presented to visualize the interplay between a grammar and the search process, which can be used for the tuning of the search algorithm.

2.2.3.1 *Simulated Annealing*

Simulated Annealing (SA) is a probabilistic, meta-heuristic method for finding the global optimum of a given function. It originates from the annealing of metals, where the cooling schedule determines if the metal solidifies into a strong, regular grid or a weak grid with many deficiencies. Every optimization step, a new design is generated from the current accepted design by applying one rule. The new design is evaluated and accepted or rejected. A design is accepted when it has a higher objective function value than the current accepted design but is also accepted with a small probability when not outperforming the current design. This mechanism allows the optimization process to escape a local maximum. The probability that a new design with a lower objective function value is accepted at time t is given by $e^{-\frac{C}{T}}$. Here, $C = m_{t+1} - m_t$, m_t is the difference between the objective function value of the current design m_t and the objective function value of the new design m_{t+1}. T is the current temperature. Every iteration, a number of moves is executed. A move consists of one rule application, followed by the evaluation of the new design and the acceptance or rejection of the new design. All moves within one iteration use the same temperature for the acceptance of designs. The temperature T is cooled down at the end of every iteration following, for example, a Cauchy cooling schedule $T_t = \frac{T_1}{1+\alpha t}$.

2.3 RELATED WORK ON CONTROL OPTIMIZATION WITHIN AUTOMATED DESIGN

In Chapter 5, a method to optimize the actuation pattern of every design candidate is sought. As will be discussed in Chapter 3, the robot designs are simulated implicitly, thus an explicit model of the dynamics of the designs is not available. This then excludes many control optimization methods.

A method that does not need the dynamics of the designs explicitly is Reinforcement Learning (RL). Reinforcement learning is a class of machine learning that optimizes a controller, also called an agent, by teaching it the desired high level behavior through rewards. Through the reward function, the designer defines the goal. At every time step t, the agents receives information about the environment in the form of an observation x_t and a reward r_t. The reward depends on the current state of the system s_t. For a fully observable environment, observation x_t and state s_t are the same. Based on the observation, the agent picks an action a_t to execute. This mapping from observation to action is called the policy $\pi(a_t|x_t)$. Formally, it is the condidtional probability of taking action a_t given the agents is in state x_t. The policy is optimized to maximize the total received reward R over the entire task. For a comprehensive overview of RL methods and the mathematical foundations we refer the reader to Sutton and Barto [70].

In recent years, Deep Reinforcement Learning (DRL) has become prominent. The field of DRL can be grouped into methods that use a discrete action space and methods that use a continuous action space. A discrete action space allows for less complex learning algorithms but suffers from the dimensionality curse more than its continuous counterpart.

An additional distinction can be made between stochastic and deterministic continuous action spaces. A deterministic policy has predictable behavior, but for some tasks a stochastic policy is desired, e.g. when the state is only partially observable. In some cases, a stochastic policy can also improve the stability of the learning process [71].

In Chapter 5,Soft Actor-Critic Reinforcement Learning (SAC) is used. SAC is an off-policy method shown to solve tasks with up to 21 actuators [71]. It is an actor-critic method for a continuous action space and it learns a stochastic policy. By leveraging the concept of maximum entropy reinforcement learning, the method learns a policy with good performance that simultaneously acts as random as possible and thereby improves robustness and exploration [72].

The control of the surfaces of the sailboat designs presented by Stump et al. [50] are learned using the proximal policy optimization implementation of Unity's ML Agent Toolkit. Unfortunately, little information is given about the tuning of the parameters and the performance of the method.

The muscle firing pattern used to actuate the robots in [28] are optimized using a GA. The variables modified by the GA are the period, phase and amplitude for each muscle. The authors state that their results are a valuable illustration of the coevolution of morphology and control, however, no information is given on the effectiveness of the actuation optimization.

2.4 SUMMARY

The chapter presents the Computational Design Synthesis framework and the literature relevant for the research conducted in this thesis. The related work on the design, fabrication and control of soft robots in section 2.1 shows that there is a need for a better design generation process, thus supporting the thesis objective as formulated in Section 1.1 of the introduction.

From the evaluation of the literature on the automated design of soft robots in section 2.2, and section 2.2.1 specifically, follows research questions 1.1 on the generation of morphologies. Section 2.2.2 illustrates the problem posed by research question 1.2 on how to simulate the generated designs for evaluation. Research question 1.3 on selecting an optimization method, needs to be answered for every optimization problem.

The first attempts at the co-design of morphology and control presented in Section 2.3 and the absence of research investigating such an approach more profoundly motivate research questions 2.1 and 2.2.

Research questions 3.1 and 3.2 formalize the need to validate the design method and allow for the comparison of the designs to existing soft robots.

3

SIMULATING SOFT BODIES

To evaluate the performance of a generated design, the designs are simulated. In this chapter, different simulation methods for the generation of simulation models of soft material are discussed to answer research question 1.2; Which simulation model best supports the evaluation of soft robots concepts for locomotion within CDS? The simulation method selected has to allow for the modeling of the designs such that these models can predict the motion of the design sufficiently accurately and simultaneously be computationally inexpensive. The aim is to identify a simulation method with the following properties:

- The simulation method has to be stable for all generated designs. This also implies that the simulation resolves collision and self-collision in a stable way.
- The method has to simulate designs without design specific tuning of simulation model parameters.
- The computational costs should be as low as possible since the method will be part of a highly iterative design process.

Three approaches are evaluated: Finite Element Method (FEM), soft body dynamics and rigid body dynamics. To select a simulation method for the evaluation stage of a CDS, using a single benchmark is not a good indication of the applicability of the method. Unfortunately, it is hard to predict which design a given method fails to model correctly and there is no single design that reveals the weaknesses of all methods. With the right tuning of simulation model parameters, most methods can simulate all the designs. However, the selected parameters for one design may not work for the next design.

The best way to expose the weaknesses of a simulation method is an automated trial-and-error approach, i.e. to use it within a CDS approach. This is demonstrated with the following example. When a simulation model fails to simulate the collision and self-collision of a design, this results in one of three cases. 1) The model can fail to resolve, e.g. a non-converging FEM step, or 2) the resolution of the collisions result in large forces and displacement that cause the design the fly away. 3) The last possibility is that the resolution of collisions results in a design that jitters and moves without being actuated. The first case, where the simulation crashes, directly provides the researcher with an example of a design that cannot be modeled properly. The other two cases result in failed models that move further and receive a higher objective function value than the correctly modeled design. Thus, the generative method actively searches for failed models. Having found a weakness, the designer mitigates the problem by tuning the simulation method parameters. This process iterates until conflicting requirements render a simulation model unfit for the use within a generative design method.

Time only permits the testing of several simulation methods and the search for a simulation method presented here is not exhaustive. In addition, since this research was conducted the field of soft material simulation has advanced.

This chapter is organized as follows. In the first section, the finite element method is discussed as a potential evaluation method. In Section 3.2, the implementation and results for soft body dynamics are presented. Finally, in Section 3.3, two rigid-body models for the simulation of soft material are presented. These two models are used in the following chapter as part of the CDS method.

A picture of a physical bending actuator with an overlay of an FEM simulation are shown in Figure 3.1 on the left. In the middle, the same bending actuator modeled with the model that will be presented in Section 3.3.1 is shown and on the right, the actuator is modeled with the model from Section 3.3.2.

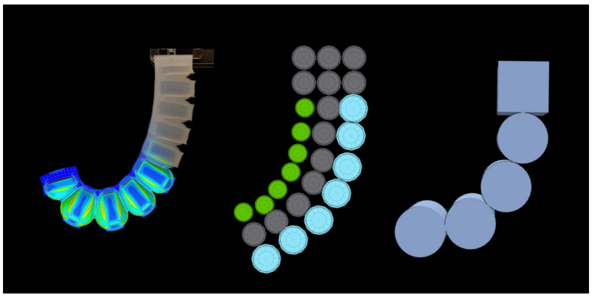


FIGURE 3.1: (left) A physical inflatable bending actuator with a partial overlay of an FEM simulation, (middle) the same actuator modeled by rows of balls interconnected with springs, (right) the same actuator modeled using hinge constraints.

3.1 FINITE ELEMENT METHOD

The finite element method is a numerical method based on locally approximating the partial differential equations that describe the process to be simulated. As shown by [73] and the work done by Dreyer as part of his bachelor thesis [74], FEM can be used to simulate the motion of a soft pneumatic actuator as a result of increased internal pressure, see Figure 3.2. Similarly, the simulation of single soft actuators with a FEM is shown by [73] and [42]. However, non-linear material stiffness and large displacements are challenging for the implicit solver used in FEM. For every time step to converge, a high mesh resolution and a small time step are needed. Without tuning and careful construction of the mesh, it is unfeasible to simulate collision and self-collision. Although the computational cost differs per implementation and model, it is generally too high for an iterative design method. As an example, the single bending actuator, without collision, simulated with Siemens NX took eight hours to solve on a Intel Xeon E3-1285Lv5 processor [74].

A simulator for soft robots based on FEM is presented by Hiller et al. [75]. Although the simulator is used for the work presented in [27] and [26], the version made available online [63] fails to simulate the few initial designs that are tried as an experiment in this work. It is decided not to investigate what the exact limitations of this simulator are and instead search for an established simulator.

The FEM implementations investigated here have high computational costs and need design-specific meshing and parameter tuning. They are likely to be able to simulate the soft robot designs given the designs are properly modeled. Currently, however, the process of modeling for a successful FEM simulation has not been automated. Therefore, it is decided that FEM-based methods are discounted as a simulation method in the generative design approach presented here. Since the field of FEM-based soft material simulation is an active research field, this conclusion should be revised in future work.

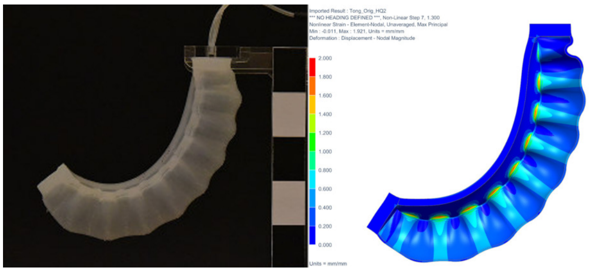

Imported Result : Tong_Orig_HQ2
*** NO HEADING DEFINED ***, Non-Linear Step 7, 1.300
Nonlinear Strain - Element-Nodal, Unaveraged, Max Principal
Min : -0.011, Max : 1.921, Units = mm/mm
Deformation : Displacement - Nodal Magnitude
2.000
1.800
1.600
1.400
1.200
1.000
0.800
0.600
0.400
0.200
0.000
Units = mm/mm

FIGURE 3.2: FEM simulation of an inflating bending actuator as part of the bachelor thesis conducted by Michael Dreyer (2016) and supervised by the author.

3.2 SOFT-BODY MODEL

Soft-body models, or spring-mass models, model a deformable body as a set of point-masses connected by springs. This model can be solved either implicitly or explicitly. Two explicit models for the simulation of soft-body dynamics are examined in this section. The springs between the point-masses are modeled as constraints. Both models accumulate forces and compute the accelerations, velocities and positions of all point-masses through an explicit time integration step. The difference between the two models, Force-Based Dynamics (FBD) and Position-Based Dynamics (PBD), is in the way constraints are satisfied.

3.2.1 *Force-Based Spring-Mass Models*

In FBD, the constraints result in forces on the point-masses to satisfy the constraints. For a spring constraint, this is the spring force itself. In the case of a point-mass violating a collision constraint, the constraint is satisfied by finding a force f_c that pushes the point-mass back in an accepted state. In this work, the Bullet Physics Library (BPL) [65] implementation is used to examine the use of FBD within a CDS. Several designs are modeled and the results are presented in Appendix A.1.

The results reveal that overshoot resulting from the explicit integration and sequential constraint solving of the method create complications with the stability of contact forces. Additionally, a small integration time-step is required to achieve deformable material that is strong enough to support its own weight. Thus, the BPL implementation of force-based spring-mass models is not tractable for the evaluation of designs within a CDS.

3.2.2 *Position-Based Soft-Body Dynamics*

In Position-Based Dynamics (PBD) as presented by [76], constraints are resolved by modifying positions of point-masses directly. A contact constraint violation, for example, is resolved by projecting the penetrating point onto the surface it penetrated, instead of forcing it out. This prevents the overshoot caused by solving collisions through constraint forces. Implementation details can be found in [77] and [78]. An extension of this simulation to also cover gases and liquids and can be found in [79].

At the time of this research, only an open-source implementation by Jan Bender [80] is available. This PBD library does not include collision detection. In [28], NVIDIA PhysX Flex is used, which also uses position-based dynamics. A beta-version is available at the time this research is conducted, however, trouble with compilation, lack of documentation and the fact that it is closed-source are deemed to make it an unreliable implementation to build a thesis on.

Attempts to add collision detection and resolution, friction and actuation to the library provided in [80] are discussed in Appendix A.2. The results show that PDB solves several of the collision resolution problems encountered in force-based dynamics. However, the adequately implementation of collision resolution is outside of the scope of this research.

3.3 RIGID BODY MODEL OF SOFT BODY SIMULATION

The problems with soft-body mass-spring models mentioned in the previous section form the motivation for modeling soft robots using rigid-body dynamics. There are many established rigid-body dynamics libraries available and additionally, rigid-bodies do not have self-collision. The rigid-body dynamics implementation used here is BPL and it uses impulse-based dynamics. In the following subsections, two different models to approximate soft material through rigid-body dynamics are discussed. Both models assume that the designs to be simulated use bending actuators like the actuator shown in Figure 3.6. The first model, a rigid balls-spring approximation, is similar to a mass-spring model, with the exception that the masses are replaced by balls. The second model, a rigid body-hinge approximation, exploits the fact that the desired behavior of the bending actuators is known further by imposing this behavior, instead of approximating the internal dynamics in the actuator. This results in decreasing computational costs and more accurate behavior, at the cost of versatility.

3.3.1 *Rigid Ball-Spring Model*

The rigid ball-spring approximation models soft material as a collection of rigid balls interconnected with springs, see Figure 3.3. A design simulated with this model is shown in Figure 3.4a. Figure 3.4b shows that every balls shares a distance constraint with its direct neighbors. By replacing the point-masses in a mass-spring model with rigid bodies, self-collision is transformed into collision between rigid bodies belonging to the same mesh. Since the rigid balls limit compression, supporting the weight of a large morphology is possible without requiring distance constraints that are too stiff for the desired behavior. Another advantage of using rigid-body dynamics is that rigid-body friction is easier to model than soft-body friction and it is readily implemented in BPL. Since constraints are still solved iteratively, a regular mesh resolution is still required. The activation of material is realized by changing the rest-length of the distance constraints. Bending is achieved through manipulating the rest-lengths of distance constraints and is illustrated in Figure 3.5. As can been seen from the figure, the size of actuated balls is changed as well. Where the rest-length of the distance constraints increases, the ball diameters are increased to prevent openings between the balls that are big enough for other balls to fit through and 'tangle up' the structure. With decreasing rest-lengths, balls are decreased to allow for the distance constraints to be satisfied.

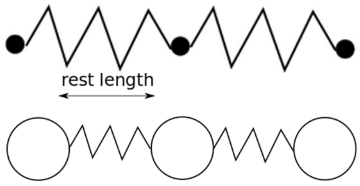


FIGURE 3.3: The spring-mass model on the top uses point-masses connected by distance constraints to represent elastic material. At the bottom, a model is shown where the point-masses are replaced by rigid bodies.

This rigid body model to approximate soft material can simulate all generated designs and it is used in the CDS approach in Chapter 4.

3.3.2 *Rigid-Body-Hinge Model*

The simulation model presented above assumes that the designs to be simulated consist of bending actuators like the actuator in Figure 3.6a. The depicted actuator is pneumatic, however, a similar bending actuation can be achieved with shape memory alloys; see Figure 3.6b. The rigid body-hinge model presented in this section exploits the assumption that only bending actuators are simulated even further to decrease computational costs.

The rigid-hinge model imposes the motion of a bending actuator as target deformation and forces are applied to the actuator such that the desired deformation is achieved. This brings three advantages. First, the actuators can be modeled with a reduced number of constraints,

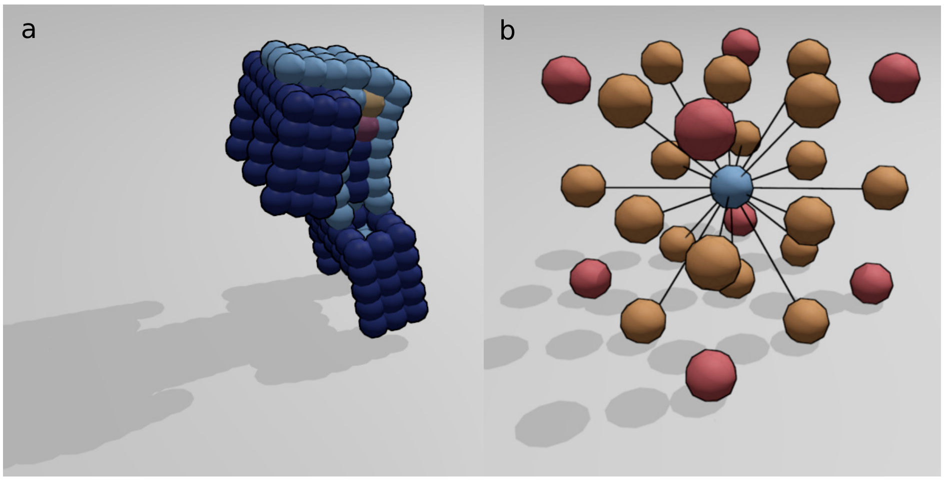


FIGURE 3.4: a) Example of a robot modeled with a ball-spring model. b) A ball and its 26 neighbors, of which 18 are direct neighbors (in orange). Only the direct neighbors are connected with distance constraints (black lines).

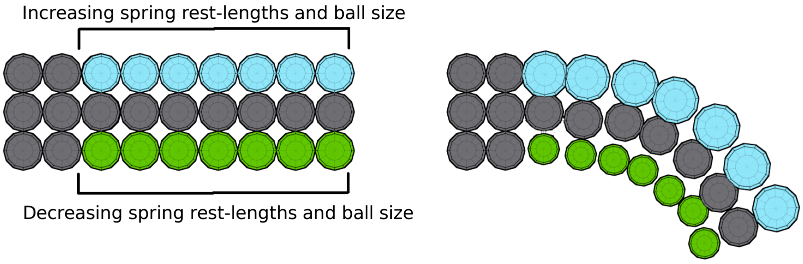

Increasing spring rest-lengths and ball size
Decreasing spring rest-lengths and ball size

FIGURE 3.5: By decreasing the springs rest-lengths and the ball size in the bottom row, and increasing them in the top row, bending is achieved.

compared to the ball-spring model from the previous section. Thus, the computational effort is reduced significantly. Second, since the actuation is imposed as desired deformation, the deformation seen in physical actuator can be approximated more accurately. Third, by modeling with actuators as units, the grid structure used before to keep the model resolution constant is no longer needed. As a consequence, shapes can be added with any rotation without having to align to a grid, see Figure 3.7.

In this model, the bending actuators are represented by a single row of rigid cylinders connected with hinge constraints. The hinge constraints in BPL are rotational spring joints. The hinges can be actuated with a servo motor constraint that can realize a desired angle. The servo motor constraints have a maximum force and velocity. Unactuated material is approximated with cubes and balls. This model is used for the evaluation of the designs generated in Chapters 5 and 6.

The reduction of the total number of constraints decreases the computational costs significantly. Additionally, tuning the hinge constraints to approximate actuator behavior directly is easier than tuning the distance constraints to produce the desired actuator behavior. The tuning of this model is discussed further in Chapter 6.

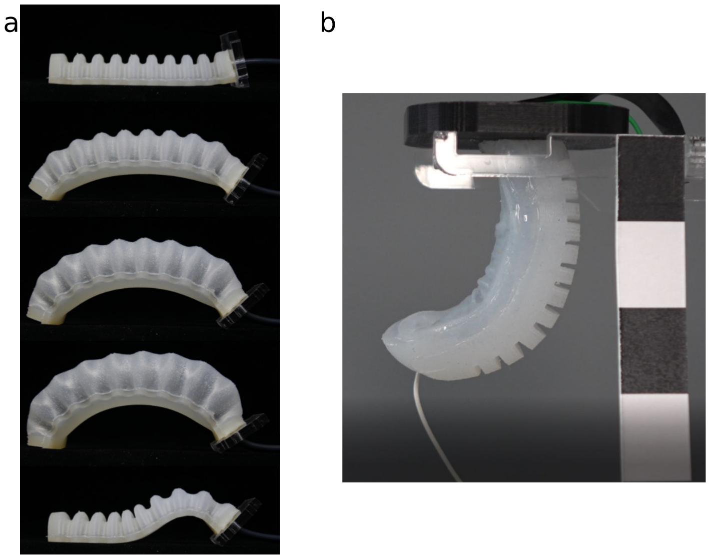


FIGURE 3.6: The bending actuator used as a model in the CDS. a) A pneumatic actuator. b) A shape memory alloy bending actuator resulting from the bachelor thesis of Matteo Tricarico, ETHZ, 2017 [81].

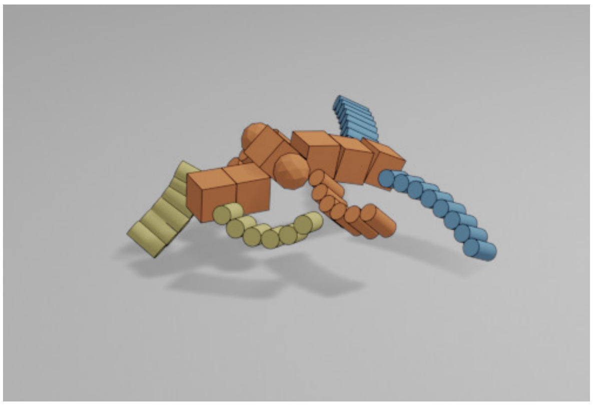


FIGURE 3.7: Example of a robot modeled with a body-hinge model.

3.4 DISCUSSION

The two models presented in Section 3.3 allow for the evaluation of design concepts but are not accurate enough for detail design. Bridging the gap between concept and physical robot still falls to the human designer and is a job that requires experience, see Chapter 6. It is also difficult to assess to what extent the model inaccuracies restrict the solution space, e.g. a CDS method using one of the rigid-body simulation models cannot produce a design with a serpentine gait since the required non-uniform friction is not modeled.

The ball-spring model has three limitations. First, the modeling of friction between the designs and the floor is limited. The balls have a small contact area with the floor, resulting in almost no friction. Experiments with other shapes, e.g. cubes, result in unnatural behavior when the shapes become 'stuck' behind each other. In the simulation model, uniform friction is implemented, rendering a serpentine gait impossible. A second complication is that when simulating a large design, the springs in the lower part of the design are not strong enough to keep the balls together. Although the rigid bodies can oppose compression forces, the springs are still subjected to extension forces. Increasing the spring stiffness increases the ability of the model to carry weight, however, it also introduces overshoot. Thus, to simulate designs with slender legs, the integration time-step in decreased and the computational increased. Third, the computational costs of this model are high since every ball can have 18 direct neighbors and 18 accompanying distance constraints; see Figure 3.4b. These constraints are solved at least once every time step, depending on the number of constraint solver iterations. The rigid-body-hinge model uses fewer constraints by modeling the actuators motion directly, at the cost of versatility. The simulation models discussed here are far removed from the physical principles of physical robots. Yet, as long as the behavior predicted by the simulation of a design is accurate enough to evaluate designs on a conceptual level, this is acceptable. In order to verify this, generated designs are prototyped and tested in Chapter 6.

3.5 SUMMARY

In this chapter, two simulation models are presented to answer research question 1.2. The models are used for the evaluation of conceptual designs in the coming chapters. The search for a simulation method conducted here is not exhaustive and the research field of soft material simulation has advanced since. A more accurate simulation model would facilitate the detailing of conceptual designs, now done by a human designer. A detailed simulation model potentially leads to increased computational costs and a staged design process could be beneficial. In a staged design process, the models presented here can be used for the CDS of the conceptual designs and a parametric optimization using a more detailed model can be used to optimize the detailed design.

The first model, the ball-spring model, approximates soft material through a grid of rigid balls connected with springs. This model is used for the evaluation of designs in Chapter 4.

The second model, the rigid body-hinge model, approximates the behavior of bending actuators more accurately, at the cost of versatility. This model is used for the evaluation of designs in Chapter 5. To verify this model is accurate enough for the evaluation of conceptual designs, generated designs are prototyped and tested in Chapter 6.

4

COMPUTATIONAL DESIGN SYNTHESIS OF SOFT ROBOT MORPHOLOGIES FOR LOCOMOTION

This chapter has been adapted from two published manuscripts, an International Design Engineering Technical Conferences & Computers and Information in Engineering Conference paper [82] and a paper published in the Journal of Mechanical Design [83].

This chapter presents two studies on the computational design of morphologies for soft robots for locomotion, each with a different spatial grammar. By examining two different spatial grammar rule-sets and two simulated annealing cooling schedules, the influence of both on the resulting designs and the optimization process is demonstrated. Both studies are composed of the following stages:

DESIGN GENERATION Each study uses a unique spatial grammar to generate the designs.

EVALUATION The designs are simulated with the simulation model described in Section 3.3.1. The fixed actuation patterns and the objective function value used in both studies are described in the following section.

OPTIMIZATION The designs are optimized using simulated annealing. However, where Study 1 uses a normal Chauchy cooling schedule, Study 2 uses Chauchy cooling with reheating.

This chapter answers research questions 1.1, 'How can a morphology generation process be created that can be controlled and guided, while still maintaining enough design freedom?' and 1.3, 'What are the characteristics of the solution space and which optimization method is suitable to carry out the optimization within CDS?'.

In the first section, the evaluation criteria and the optimization method used in both studies are presented. The two following sections present the two studies. The chapter is finalized with a short summary.

4.1 EVALUATION AND OPTIMIZATION

To obtain the objective function value for a design, it is simulated for 15 simulation-seconds using the ball-spring simulation model discussed in Section 3.3.1. As described by the task definition in Section 1.3, the objective is to maximize the minimum distance traveled by any ball of the design, while minimizing the average loss of potential energy. Penalizing the loss of potential energy prevents the evolution of designs that move by falling. The unconstrained multi-objective optimization problem can be stated as:

$$\underset{\mathcal{D}}{\text{maximize}} \left(\min_i |\mathbf{x}_i^{t_0} - \mathbf{x}_i^{t_{end}}| \right), \ i = 1, \ldots, N \tag{4.1}$$

$$\underset{\mathcal{D}}{\text{minimize}} \left(\max \left(\frac{1}{N} \sum_{i=1}^{N} (y_i^{t_0} - y_i^{t_{end}}), 0 \right) \right) \tag{4.2}$$

Here, $\mathcal{D}$ is a design consisting of a set of balls, N is the number of balls in the evaluated design $\mathcal{D}$ and $\mathbf{x}_i$ a two dimensional vector with the x- and z-position of ball i. The time superscripts t_0 and t_{end} stand for the start and end of the simulation-time. The y-position (height) of ball i is denoted as y_i. Equation 4.1 maximizes the shortest distance traveled in the xz-plane by any ball during the simulation. Selecting the least moving ball prevents a high objective function value for designs that merely rotate. Equation 4.2 minimizes the loss of potential energy. This is a penalty on the use of initial potential energy to cover distance. The max-function in Equation 4.2 makes the problem ambivalent towards gaining potential energy. The multi-objective optimization problem is optimized as a weighted sum with both weights set to 1. The objective function value of a design $\mathcal{D}$ to be maximized is given by:

$$f(\mathcal{D}) = \min_i |\mathbf{x}_i^{t_0} - \mathbf{x}_i^{t_{end}}| - \max \left(\frac{1}{N} \sum_{i=1}^{N} y_i^{t_0} - y_i^{t_{end}}, 0 \right) \tag{4.3}$$

A design can have a radically different gait with only a small change to its morphology and therefore, the solution space is discrete. As is examined in Section 4.2.3, the spatial grammar rules presented below add and remove large amounts of material in one rule application and thereby cause large jumps in objective function value. Additionally, it is expected there exist multiple morphologies and gaits that exhibit locomotion, thus the solution space is expected to be multi-modal. As discussed in Section 2.2.3 of the background, this is the motivation for the selection of simulated annealing for the optimization of the designs.

To find the initial temperature for simulated annealing, a random walk is performed. The standard deviation of the obtained objective values is set as initial temperature since it is a measure of the expected improvement per move, see [84]. An optimization run consists of 150 iterations, with 50 moves per iteration. A Cauchy cooling schedule is used, i.e. $T_t = \frac{T_1}{1+\alpha t}$. A cooling rate α of 0.2 is used.

During evaluation in simulation, the actuators of the design execute a predefined actuation pattern, effectively making the actuation pattern a constraint of the optimization problem. The actuation patterns are assigned to actuators by the spatial grammar rules during generation, i.e. the actuation pattern is a parameter of the parametric spatial grammar rules. Four phase-shifted cyclic patterns, shown in Figure 4.1, enable agonist-antagonist behavior. The patterns overlap to facilitate sequential activation, e.g. the lifting of a leg before it is moved forward.

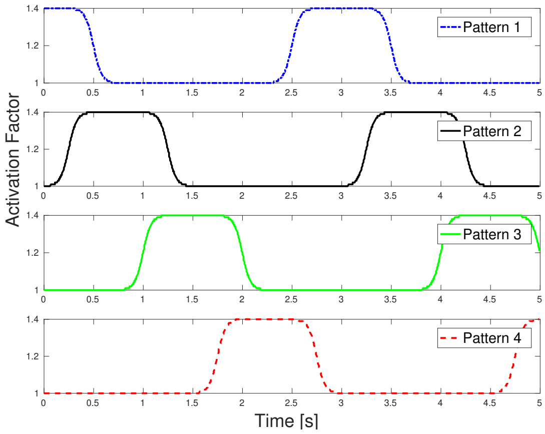


FIGURE 4.1: The four patterns used for activation.

4.2 STUDY 1

In this section, the spatial grammar of the first study is presented, as well as the results, an analysis of the grammar performance and a brief discussion.

4.2.1 *Spatial Grammar 1*

A spatial grammar is designed to generate locomotive soft robots. Calisti et al. [85] distinguish five fundamental types of locomotion: crawling, legged locomotion, jumping, flying and swimming. Because of the different simulation models involved in the latter two, only the first three types are considered. The rules presented here guide the generation towards morphologies that perform these types of locomotion. Before presenting the actual grammar rules, the assumptions concerning three aspects of the design environment are explained: The grammar abstraction level, the activation type and the 3D design space.

First, the abstraction level of the grammar is defined. Rules can be designed to manipulate unit cells of a design, e.g. units of silicone material, but rules can also be designed to manipulate assemblies of a design, e.g. a leg or an arm. Whereas a low abstraction level allows for large design freedom, a high diversity of results and little bias, it increases the size of the optimization problem and makes it harder to use preexisting knowledge in the design process. Considering the vast design space and the resource intensive simulation, priority is given to guidance, i.e. using domain specific guidance.

Second, an activation type is chosen. Existing soft robots use shape memory alloys, tendons or inflation to generate motion. These physical activation methods are used to bend and/or extend parts of a robot. The actuators used in this study are designed with the actuation principle of inflating bending actuators. However, other physical activation methods could be used to realize the motion. As is briefly mentioned in Section 3.2.1, the generated robots are actuated with bending actuators, as opposed to extending actuators. The majority of the existing soft robots discussed in Section 2.1 use bending actuators.

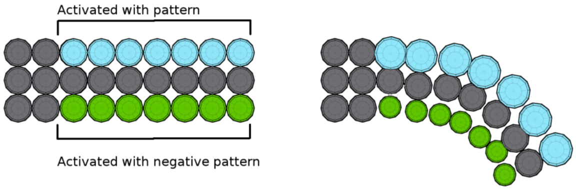


Figure 4.2: Bending is achieved by increasing the length of the springs between balls on one side, here the top row, and decreasing the rest length on the other side, i.e. the bottom row. In simulation, the size of the balls is changed accordingly to keep balls from one row passing through the gaps of the neighboring row.

Third, the nature of the design space is explained. Designs consist of rigid balls connected with springs. The maximum space a design can occupy consists of 15x15x15 balls. Unit cells possess the following characteristics: a friction coefficient, a material stiffness that governs the spring properties of the connections with neighboring balls, and an activation pattern. The activation pattern is applied to the rest length of the springs and thereby prescribes the desired distance between the balls. The unit balls form assemblies that are the building blocks used by the grammar. In the generation process, however, assemblies are allowed to overlap and can also be partially deleted. Bending assemblies are realized by a composition of rows of balls. By expanding and contracting the rows, a bending motion is generated, see Figure 4.2. The generation process starts with a completely empty design space and follows by applying spatial grammar rules.

A rule set of seven rules is developed that are aimed at generating designs that crawl, hop and walk. There are specific rules devised to support crawling and walking. Hopping, however, is less a result of the shape of a morphology and more the result of the ratio between the mass and the activation of the morphology. In Figure 4.3, the seven rules of the spatial grammar are shown. The left column shows the left hand side, the middle column shows the right hand side and the last column shows the required parameters. The rules are described as follows:

- Rule 0 creates the initial shape and takes no parameters.

- Rule 1 adds a bending appendage in the Z-direction, as seen in crawling robots. The activation pattern and the material stiffness are parameters for the rule application. The arrows in Figure 4.3 show the bending direction of the appendage when activated and the blue balls have a higher friction coefficient than the white ones.

- Rule 2 is a bending appendage in the Y-direction and is expected to assist legged locomotion. Rule 2 also takes an activation pattern and a material stiffness as parameters. The assemblies of Rule 1 and Rule 2 together are anticipated to form a leg. Here, the assembly from Rule 1 would behave as the upper part of a leg and the assembly from rule 2 the lower part.

- Rule 3 is a gate shaped assembly that is unactuated and can be oriented in two ways. The objective of this rule is to lift the design off the ground and reduce the friction by reducing the contact area with the floor. However, the rule can be applied anywhere in the design, not only at the bottom.

- Rule 4 is a bending appendage with high friction at the end. It takes an activation pattern, a material stiffness, an activation direction and orientation as parameters. A positive activation direction results in a concave leg shape when activated. Exploring the design space through manual design showed the benefit of diagonally attached legs for the stability of the robots. Thus, this rule adds a diagonal appendage to the morphology. Additionally, this rule introduces actuation in the Y-plane.
- Rule 5 removes balls from the design. The removed area is box shaped and the dimensions are parameters. This rule enables the partial deletion of assemblies to give them a new behavior and limits the size of the designs.
- Rule 6 changes the characteristics, namely activation pattern and material stiffness, of all neighboring balls to match the characteristics of the ball the rule is applied to. After multiple rule applications, it is likely that a design consists of the remnants of many partial assemblies. The objective of this rule is to homogenize the morphology and balance chaos naturally building over the generations.

Since assemblies are allowed to overlap and Rule 5 and 6 can disrupt assemblies, a high level of generation freedom is preserved. Rules are applied randomly as long as the size of the design is within set parameters, the deletion rule is invoked otherwise. The rules are applied with a random material stiffness value and a randomly selected activation pattern. In case a rule tries to build outside of the design space, the added assembly is cut to fit.

Inspired by nature and engineering alike, the generation of symmetric designs can be imposed. This is achieved by applying each rule on both sides of the X-axis of a design. The grammar is investigated with and without symmetry imposed.

To illustrate the use of the spatial grammar, Figure 4.4a shows the generation a four-legged design created by hand picking an applying rules. The evaluation of the resulting design is shown in Figure 4.4b. An overview of the iterative design process using this spatial grammar and simulated annealing is given in Figure 4.5

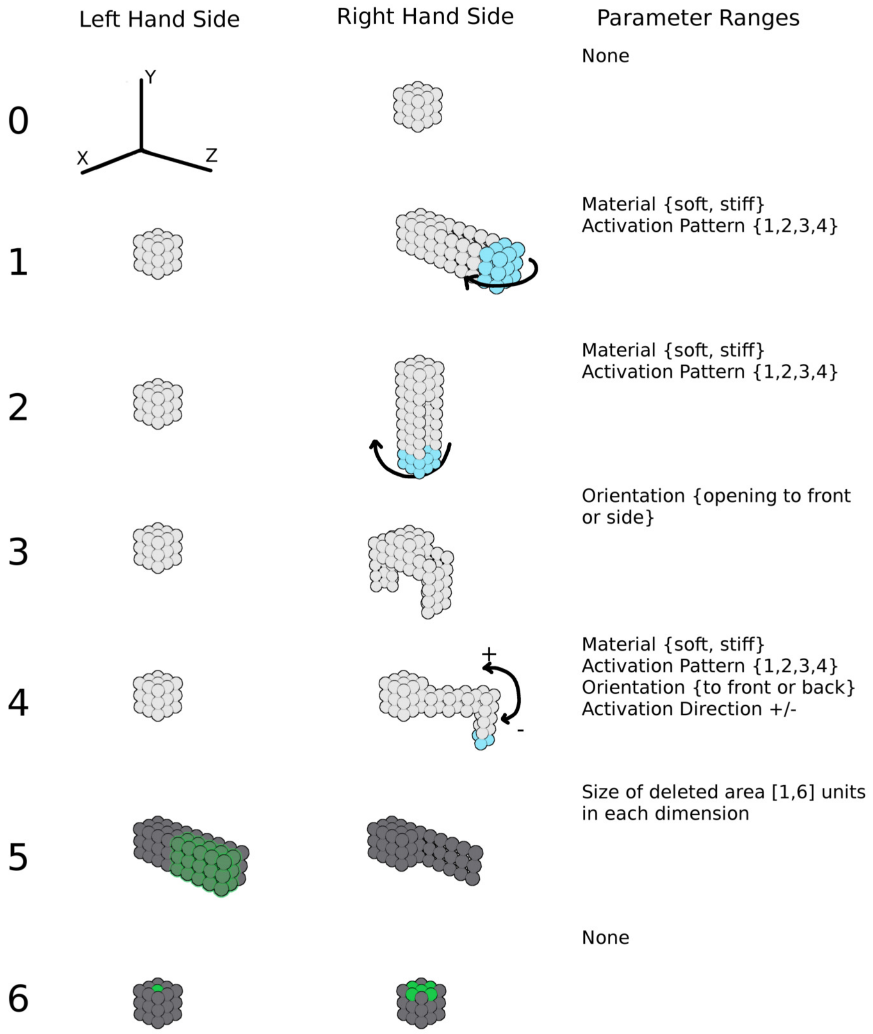

Left Hand Side
Right Hand Side
Parameter Ranges
0
Y
X
Z
None
1
Material {soft, stiff}
Activation Pattern {1,2,3,4}
2
Material {soft, stiff}
Activation Pattern {1,2,3,4}
3
Orientation {opening to front or side}
4
+
-
Material {soft, stiff}
Activation Pattern {1,2,3,4}
Orientation {to front or back}
Activation Direction +/-
5
Size of deleted area [1,6] units in each dimension
6
None

FIGURE 4.3: The seven rules of the ruleset. The first column shows the left hand side, the second column the right hand side. The third column shows the required parameters and their range. Blue balls (found in the extremities of rule 1, 2 and 4) have an increased friction coefficient. Bending arrows show the bending direction of the actuators.

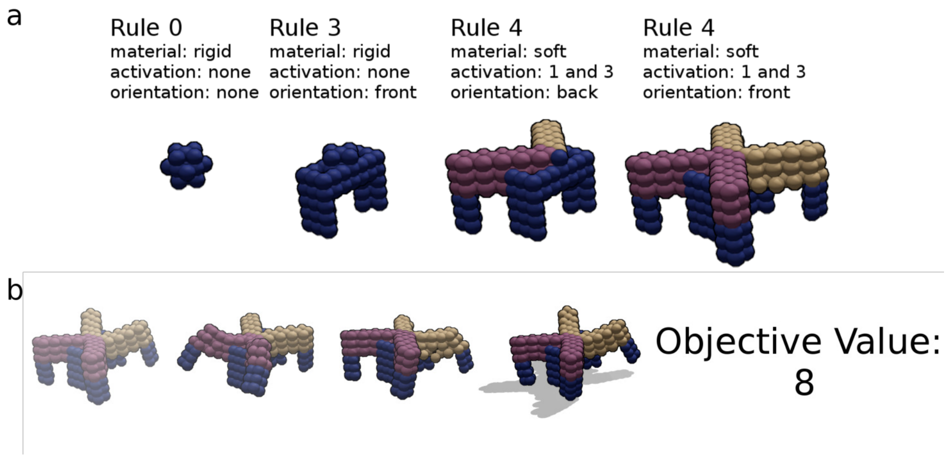


FIGURE 4.4: a) The generation of a design created by hand-picking and applying rules. For every application, the rule and the used parameters are given. b) The motion and objective function value of the hand-built design.

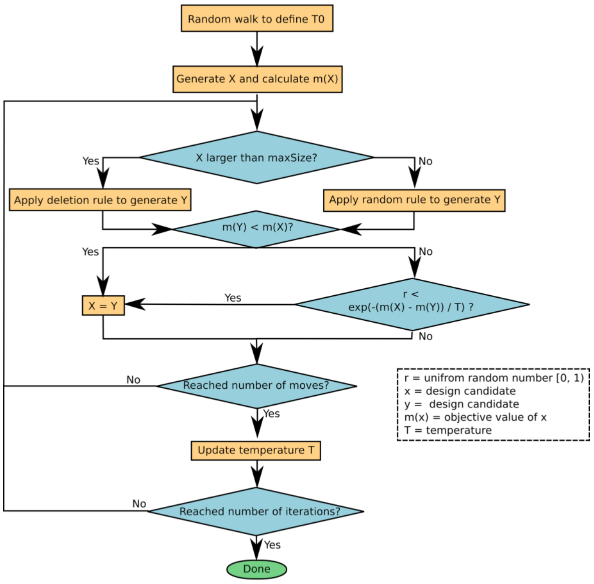

Random walk to define T0
Generate X and calculate m(X)
X larger than maxSize?
Yes
No
Apply deletion rule to generate Y
Apply random rule to generate Y
m(Y) < m(X)?
Yes
No
X = Y
Yes
r <
exp(-(m(X) - m(Y)) / T) ?
No
Reached number of moves?
No
Yes
r = unifrom random number [0, 1)
x = design candidate
y = design candidate
m(x) = objective value of x
T = temperature
Update temperature T
Reached number of iterations?
No
Yes
Done

FIGURE 4.5: A flowchart of the optimization loop.

4.2.2 *Results*

The first row in Table 4.1 shows the results of simulated annealing runs of 150 iterations, with 50 moves per iteration. The average objective value of 31 runs is 37.2, with a standard deviation of 10.4, as shown in Figure 4.6. The minimum final objective function is 18.3, the maximum 64.8. The average runtime on an Intel Xeon E3-1285Lv5 processor is 1045 minutes. The convergence of the runs is discussed in the discussion section.

As is shown in Figure 4.7, the average objective value of 40 runs with 100 iterations and 25 moves is 32.9, with a standard deviation of 9.9. The minimum final objective function is 10.3, the maximum 58.4 and the average runtime 852 minutes. Although the average value of the shorter runs is well below that of the longer runs, the maximum values are very close.

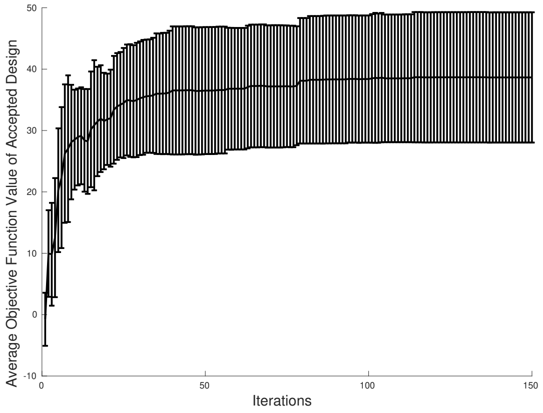


FIGURE 4.6: The average convergence and standard deviation of 31 runs of 150 iterations with 50 moves.

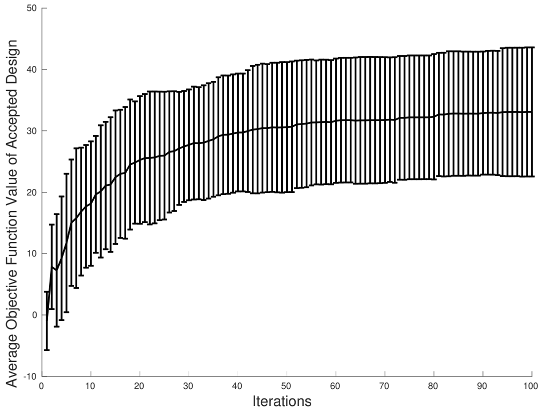


FIGURE 4.7: The average convergence and standard deviation of 40 runs of 100 iterations with 25 moves.

In Figures 4.8 to 4.10, designs resulting from the optimization process are shown. The different colors of the balls represent the different activation patterns and material stiffnesses.

TABLE 4.1: Performance measures for different optimization processes. It shows that long runs have better results on average. However, the difference between the maximum values is small.

Process	# Runs	Average Best	Min	Max	Std
150 Iterations, 50 Moves,	31	37.2	18.3	64.8	10.4
100 Iterations, 25 Moves	40	32.9	10.3	58.4	9.9

The dark blue balls have a rigid connection to their neighbors, all others have an elastic connection. The different friction values are not visible in this picture.

The generated results are manually classified as walker, hopper or crawler. Figure 4.8 shows symmetric designs demonstrating a hopping and a crawling gait. In Figure 4.9, symmetric designs exhibiting legged locomotion are shown. Figure 4.10 shows asymmetric walking, hopping and crawling. Although hopping and crawling designs are found for both symmetric and asymmetric grammars, asymmetric designs showing legged locomotion are rarely found. Design D in Figure 4.10 is symmetric, although the symmetry was not imposed during design generation, it evolved naturally. The occurrence of the three types of gaits and their performance is listed in Table 4.2. It can be seen that crawlers and hoppers are more common than walkers. These two types contain the best individuals but the design quality is less consistent.

Since it is hard to capture the gaits of the designs on paper, a video showing the locomotion of the presented designs can be found here: https://youtu.be/C43uFq4B8FU.

TABLE 4.2: The occurrence of the different classes, in a set of 105 manually classified designs, and their performance measures.

Measure	Walkers	Hoppers	Crawlers
Occurrence	16%	38%	46%
Average value	33.3	34.2	31.5
Min value	21.2	19.9	10.3
Max value	48.9	64.8	58.4
Std value	7.1	10.4	9.9

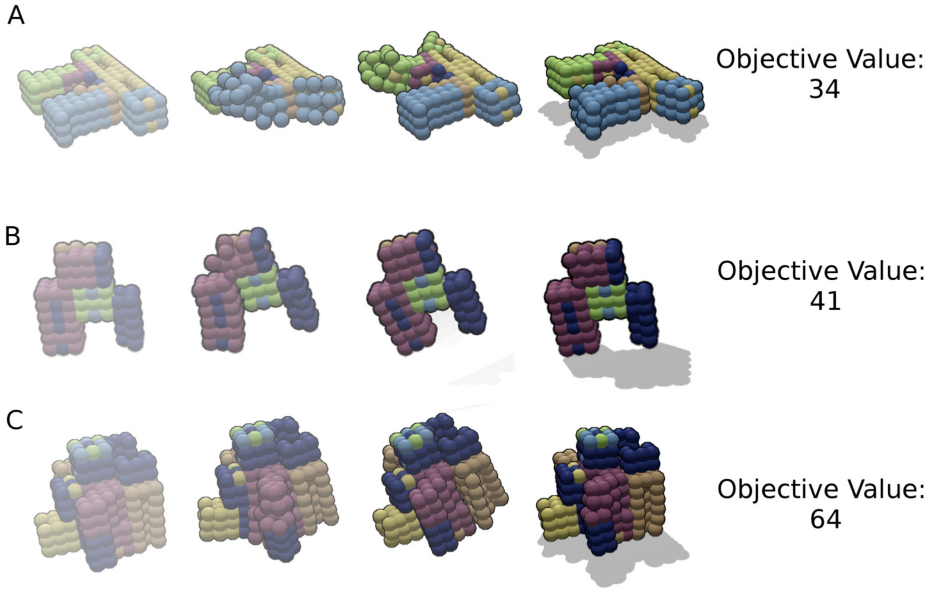


FIGURE 4.8: Symmetric designs showing hopping and crawling locomotion. Design A crawls using the impulse from throwing its backside extension up. Design B uses the bending of its back vertical extension to hop forward. Design C hops forward by lifting off and rotating before landing.

4.2.3 *Grammar Performance*

The performance of the spatial grammar is measured by the following three criteria: the predictability of short-term rule application quality, the predictability of long-term rule application quality and the anticipated behavior of assemblies.

4.2.3.1 *The predictability of short-term rule application quality*

The predictability of short-term rule application quality is defined as the direct effect of a rule application on the objective function value of the current design. In Figure 4.11, the change in objective function value following a rule application is plotted for all rules. It shows that rules have, on average, a negative effect on the objective function value. Although a small percentage of rule applications do improve the designs, the effect of a single rule application is unpredictable. The effect of a rule cannot be assessed independently of the design it is applied to, the location where it is applied and the stiffness and activation variables. Except for the deletion rule, attempts to find a correlation between the success of a rule application and the current design size were unsuccessful.

4.2.3.2 *The predictability of rule application on long term quality*

To evaluate the predictability of the long term quality of a rule application, the occurrence of rules in the history of a final designs is measured. A rule can have a negative immediate effect on the objective function value but can be a necessary intermediate step towards

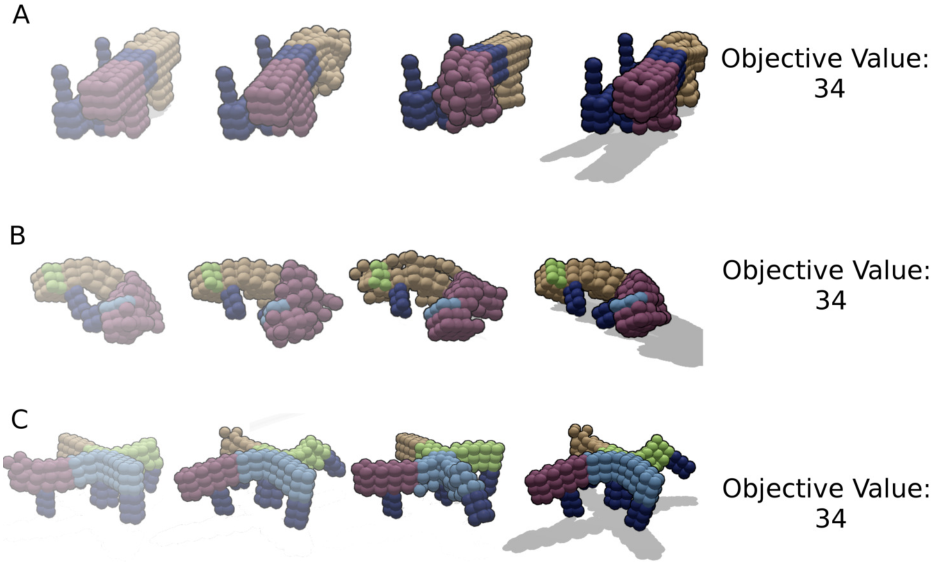


FIGURE 4.9: Symmetric designs showing legged locomotion. By lifting and bending one side at the time and using the opposite side as pivot point, Design A moves forward. Design B pushes itself forward with its alternating back legs (dark blue extensions). Design C has a four-legged gait.

a better design. Table 4.3 shows the occurrence of rules in the history of final designs. It is generated by collecting all accepted rule applications. It should be noted that the deletion rule has a disproportional probability of being applied since after a certain number of additive rule applications, it is automatically invoked to limit the size of the designs. Whereas Figure 4.11 shows that the average effect of the Y-extension rule is worse then that of the other rules, Table 4.3 shows it is the most successful building block. Figure 4.11 suggests the locality rule performs well but its relevance is marginal.

4.2.3.3 *The anticipated behavior of sub-assemblies*

The behavior of the different assemblies in the designs is assessed. The Z-extension rule is found functioning as anticipated in several final designs, e.g. Figure 4.9 A and Figure 4.8 A. The deletion rule has kept the average design size from growing larger than 10% of the maximum design size, which would be 15x15x15. The gate rule was intended to stably raise a design off the floor. Design B in Figure 4.8 has a front support that originates from a gate rule application. A full, unaltered gate assembly is rare within the final designs. The Y-extension rule is recognizable in many designs. Design D in Figure 4.10 uses it as support, whereas B and C in Figure 4.8 use it as means to hop forward. The diagonal-leg rule works as intended and is demonstrated by design C in Figure 4.9 and design D in Figure 4.10. However, V-shaped design (B) in Figure 4.9 uses the legs to push itself forward. Variations on this V-shaped design occur very frequently in the set of final designs. The effect of the rule that improves locality is hard to determine. The need for it, however, is clear. Many

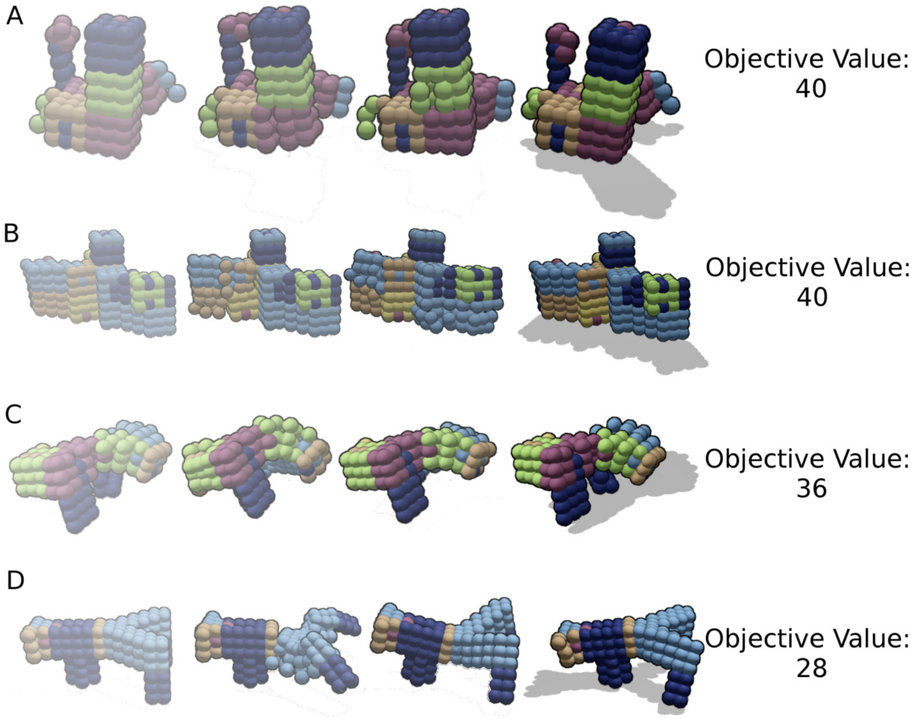


FIGURE 4.10: Asymmetric designs showing legged, hopping and crawling locomotion. Design A hops. Design B crawls with a motion that resembles the motion of a caterpillar. Design C shows two-legged locomotion (the two dark blue parts are the legs) and has a dynamic gait. Design D uses symmetric diagonal legs. Here, the symmetry evolved naturally and was not enforced during design generation.

designs have over 20 separate clusters of activated material. This is not only likely to be inefficient, it scales up the control problem and makes fabrication more difficult.

The symmetry constraint is used for the generation of half of the results. Comparing the performance of all symmetric results with all asymmetric results, the symmetric results have a better overall performance. The difference however, is smaller than the standard deviation of the combined asymmetrical results, i.e. it is not significant.

For this grammar, the change introduced by a single rule application is independent of the current design. The average assembly added by a rule consists of 70 unit balls. For an average design size of 250 balls, this means a change of 28%.

4.2.4 *Discussion Study 1*

The presented method for the automated design of virtual soft robots uses a spatial grammar combined with simulated annealing. A large variety of successful gaits result from the generation process without explicitly being programmed. Examples of all three locomotion principles, namely walking, hopping and crawling, can be found in the solution set. The method also produces various designs with appendages. The difference in occurrence

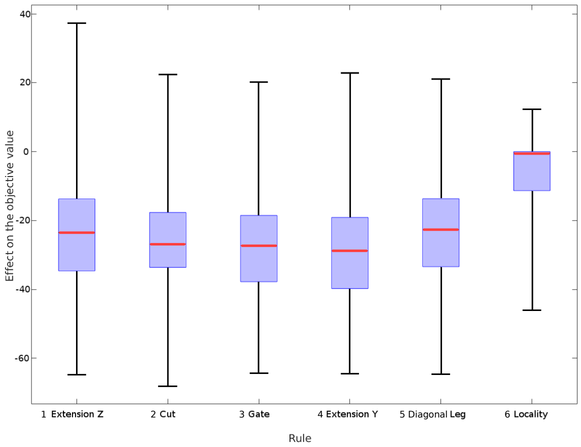


FIGURE 4.11: Average improvement of the objective value for the different rules. All rules have, on average, a negative effect on the objective value, but a small percentage of rule applications improve the design. The median is shown as a line, the standard deviation as a box.

between the three types can be explained by the fact that the definition of walking is narrower than that of the other two. Additional, the generation of a walker requires more rule application and is therefore less likely and the four actuation patterns limit the number of possible walking designs.

The presented convergence results show that this is a hard problem to optimize. The standard deviation of the final objective values is large. Two causes are discussed here.

First, it is very likely, but hard to show, that the solution space contains many local maxima. Adding a small amount of material has shown to be the difference between walking and falling, i.e. neighboring designs do not have similar objective function values. The rules trigger large objective value changes, as can be seen in Figure 4.11.

Second, since the absolute change in shape introduced by a single rule application is independent of the current design, the size of the simulated annealing neighborhood is relative. Whereas large designs require multiple rule applications to change the shape significantly, small designs require only one. Escaping a local maximum is therefore easier for a small design than it is for a large one. This poses a problem for the tuning of the cooling schedule. Using this plain simulated annealing temperature schedule, the premature convergence of individual optimization runs would not be solved by running longer. In Study 2, presented in the next section, reheating of the simulated annealing process shows to facilitate the escape of local maxima.

By using a spatial grammar for design generation, a part of the design requirements and constraints are modeled in the generation method, rather than the objective value function. The design generation explicitly guides the type of morphologies that are generated. The grammar rules are used both in expected and unexpected ways and many designs have appendages. The analysis of the rules provides insight into the performance of the individual grammar rules and suggests the following improvements.

TABLE 4.3: Occurrence of rules in the history of final designs. Note: when designs exceed the maximum size, the deletion rule is selected.

Rule	Occurence %
Extension Z	13.69
Deletion	34.95
Gate	15.29
Extension Y	19.12
Diagonal Leg	13.58
Locality	3.37

First, a variable rule size will enable changes relative to the current design size and current temperature. The challenge lies in maintaining the desired behavior for the scaled rules.

Second, the added assemblies are disrupted by the deletion rule and by overlapping assemblies. Partial deletion of assemblies and overlapping assemblies allow the generation of shapes that are not a combination of intact assemblies. This expands the design space, but it also interferes with the modeled purpose of the rules. Since assemblies are added and partially deleted every iteration, longer optimization runs show a higher degree of disarray and a larger number of separate activation clusters. The impact of the proposed homogeneity rule does not suffice to prevent this. Study 2, presented in the next section, presents a spatial grammar that preserves the assemblies.

Third, detailing the left-hand side of the rules further will increase the control over the generated morphology types. This would allow for more control over the generation process and could be used to successfully counter the increasing disarray or increasing the percentage of walkers.

A subset of the resulting designs has the potential to be fabricated with a few conventional soft actuators and serve as conceptual designs.

Summarizing, Study 1 shows the feasibility of the CDS of soft robots for locomotion but also suggests several improvements. The two most important ones are a grammar that preserves assemblies and a cooling schedule with reheating to escape local maxima, both of which are implemented in Study 2. Also, the grammar in the Study 2 is only used to generate symmetric designs.

4.3 STUDY 2

Based on the results and the spatial grammar analysis from the first study, changes to the optimization and the spatial grammar are made for the second study.

4.3.1 *Spatial Grammar 2*

The spatial grammar presented in this section relies on the same assumptions as the grammar of Study 1, presented in Section 4.2.1. However, the generation presented in this section is constrained to symmetric designs. Also, the deletion rule (Rule 6) is replaced by an undo rule. The undo rule reverts a single additional rule application completely.

Again, a rule set of seven rules is formulated and shown in Figure 4.12. Here, the changes made to the grammar of study 1 are presented.

Rule 0 creates a smaller initial shape than before so small designs do not have to start with bulk.

Rule 1, 2, and 3 add bending extensions in the Z-, Y- and X-directions. Like before, the material stiffness and actuation pattern are rule parameters. Here, however, three parameters are added. First, an offset defines which ball of the new extension will be used as origin. The origin ball will be located at the location of the ball the rule is applied to. Second, the activation direction defines if the extension bends in the positive or the negative direction. The last parameter defines if the extension is pointing up or down or to the front or to the back. This parameter does not exist for Rule 1 since rules are applied symmetrically around the X-axis. The layers of balls at the end of the extensions consist of balls with a higher friction coefficient (shown in blue).

The gait shape and the diagonal leg of the previous grammar are preserved in Rules 4 and 5.

Rule 6 removes balls from the design by undoing a previous additive rule application. A rule application from the history of the current design is chosen at random. After it is assessed that undoing this rule does not result in the morphology splitting, the rule is undone. By invoking this rule when designs reach 800 balls or 12 separate clusters of material, the rule is used to limit the size and the number of separate actuators of the designs.

The added assemblies are allowed to overlap. Rules are applied randomly as long as the size of the design is within the set parameters, the undo-rule is invoked otherwise. The rules are applied with random parameters. In case the assembly of an additive rule partly falls outside the design space, the added assembly is cut to fit.

To illustrate the method, the generation of a design created by hand-picking and applying rules is shown in Figure 4.13a. The motion and objective function value of the hand-built design shown is shown in Figure 4.13b.

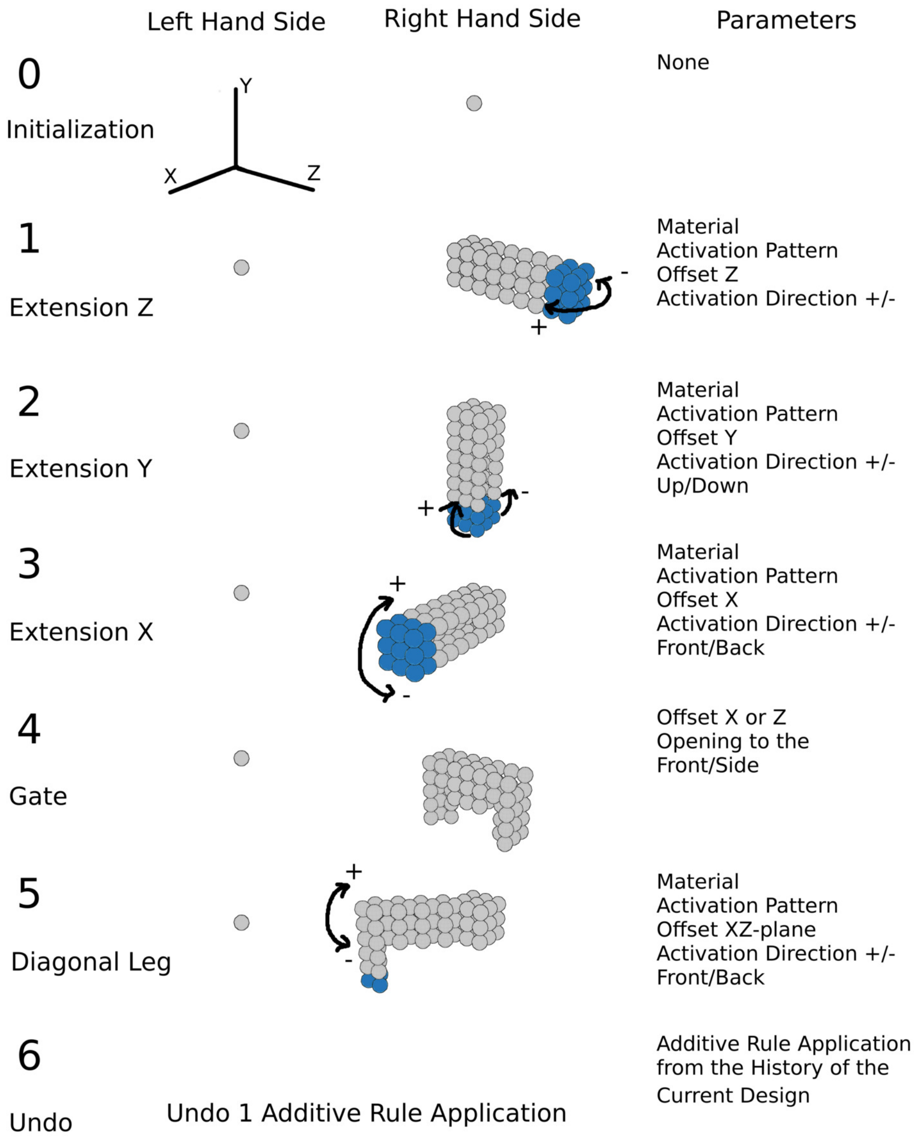


FIGURE 4.12: The seven rules of the rule set. Rules 0 to 5 add material, Rule 6 removes material. The rule parameters are listed in the right column, the activation directions are shown by the arrows. Dark blue balls have a higher friction value than the light grey balls.

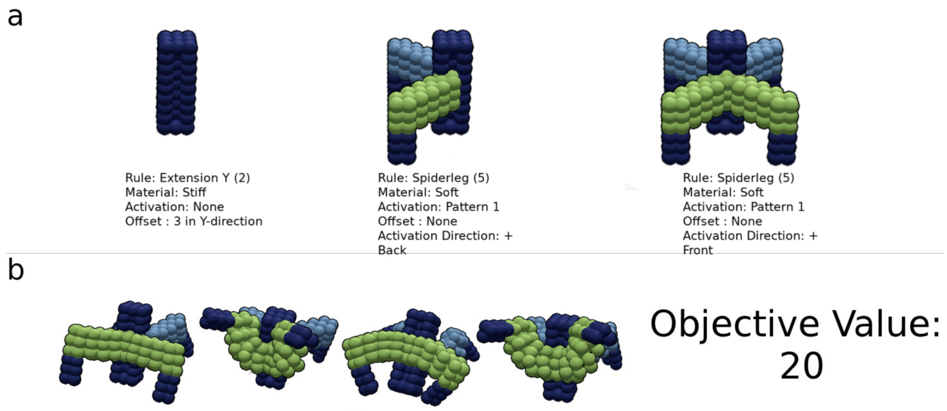


FIGURE 4.13: a) The generation of a design created by hand-picking and applying rules. For every application, the rule and the used parameters are given. b) The motion and objective function value of the hand-built design.

4.3.2 *Optimization*

To prevent premature convergence, reheating is implemented. The convergence plots from Section 4.2 show that, without reheating, the optimization process stagnates, on average, around iteration 80. Increasing the temperature does not prevent it since the neighborhoods of large and small designs are not equal. For a large design to change its shape, multiple rule applications need to be accepted. When the current design is a local maximum, a high temperature is required to escape. For a smaller design, the high temperature prevents convergence. Therefore, tuning of the initial temperature and cooling rate is not enough to achieve a cooling scheme that is successful for both large and small design solutions. A reheating scheme is described by Triki et al. [84], however, it requires the move size to be variable. Unfortunately, within the spatial grammar presented here, a small or large rule application has no meaning.

The following implementation is used. When no new solution is found in the last five iterations, the temperature is raised by 0.1. The resulting simulated annealing algorithm is given in Algorithm 1.

4.3.3 *Results*

The simulated annealing runs have 150 iterations, with 50 moves per iteration. The results of 24 runs are reported in Table 4.4. The table also shows the results for 24 runs without reheating and the results for 24 runs with reheating and 250 iterations. Figures 4.14, 4.15 and 4.16 show the respective convergence plots. As can be seen from the table as well as the plots, the processes without reheating are trapped in a local maximum. Although none of the converge plots show a converged process within the number of executed iterations, the processes with reheating still improve. The average temperature versus iterations for both with and without reheating is shown in Figure 4.17a. The corresponding acceptance rate is shown in Figure 4.17b. Around iteration 100, the temperature of the process with reheating starts to increase. The average objective function value of the accepted solutions does not improve anymore but the average objective function value of the best solutions does. The runs with 250 iterations show similar results. The average runtime for 150 iterations on an Intel Xeon E3-1285Lv5 processor is 1882 minutes (31.4 hours). In Figure 4.18, the average objective function value for designs of different sizes is shown. Designs with a number of balls ranging between 300 and 610 are more likely to have a good objective function value than designs that fall outside this range. Note that this result only holds for the current grammar and optimization model.

In Figures 4.19 to 4.21, designs resulting from the optimization process are presented. Figure 4.19 shows designs demonstrating a hopping gait. In Figure 4.20, walking designs are shown. Figure 4.21 shows a crawling and a rolling design. A video showing the locomotion of the presented designs can be found here: https://youtu.be/zMgdI3MNuA0.

Algorithm 1 Simulated Annealing

```
Require: $T_{start}$, cooling rate $\alpha$, reheating $r$, number of iterations $I$, number of moves $M$,
    maximum design size $maxDesignSize$, stagnation threshold $stagnation$
Ensure: Best found solution $S_{best}$
$\mathcal{D}_{curr} \leftarrow initialDesign()$
$\mathcal{D}_{best} \leftarrow initialDesign()$
$noImprovement \leftarrow 0$
for $i = 0$ to $I - 1$ do
    for $m = 0$ to $M - 1$ do
        $noRuleApplied \leftarrow 1$
        while $noRuleApplied$ do
            if $size(\mathcal{D}_{curr}) > maxDesignSize$ then
                pick $undoRule$
            else
                randomly pick $rule$
            end if
            randomly pick $parameters$
            $\mathcal{D}_{new} \leftarrow applyRule(\mathcal{D}_{curr}, parameters)$
            if $succesfulRuleApplication$ then
                $noRuleApplied \leftarrow 0$
            end if
        end while
        if $f(\mathcal{D}_{new}) < f(\mathcal{D}_{curr})$ then
            $\mathcal{D}_{curr} \leftarrow \mathcal{D}_{new}$
            $\mathcal{D}_{best} \leftarrow \mathcal{D}_{new}$
            $noImprovement \leftarrow 0$
        else if $\exp\left(\frac{f(\mathcal{D}_{new}) - f(\mathcal{D}_{curr})}{T_i}\right) > rand()$ then
            $\mathcal{D}_{curr} \leftarrow \mathcal{D}_{new}$
            $noImprovement \leftarrow 0$
        else
            $noImprovement \leftarrow noImprovement + 1$
        end if
    end for
    if $noImprovement < stagnation$ then
        $T_{i+1} = \frac{T_{start}}{1 + \alpha i}$
    else
        $T_{i+1} = min(T_i + r, T_{start})$
    end if
end for
return $\mathcal{D}_{best}$
```

Process	Average Best	Minimum	Maximum	Standard deviation
150 iteration reheating	43.9	13.8	68.9	16.5
150 iteration no reheating	37.4	12.5	76.5	18.3
250 iteration reheating	52.0	14.8	93.7	17.2

Table 4.4: Results for 3 optimization processes, each consisting of 24 runs. The first column describes the parameters used for the sets of 24 runs. In the subsequent three columns, the average best objective function value, the minimum objective function value and the maximum objective function value are reported. The last column shows the standard deviation of the best objective function values.

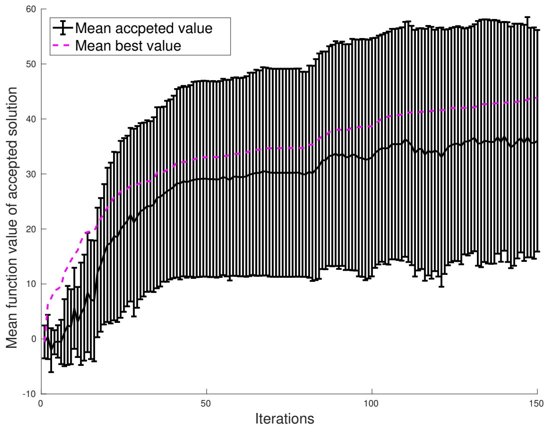


Figure 4.14: The average accepted objective function value and the average best objective function value of 24 runs with 150 iteration and 50 moves with reheating. The standard deviation of the average accepted value is shown.

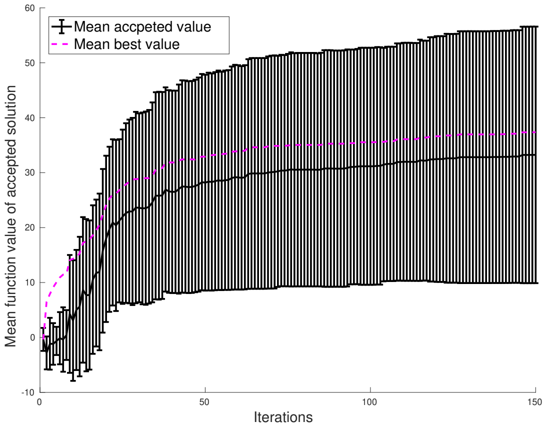


Figure 4.15: The average accepted objective function value and the average best objective function value of 24 runs with 150 iteration and 50 moves without reheating. The standard deviation of the average accepted value is shown.

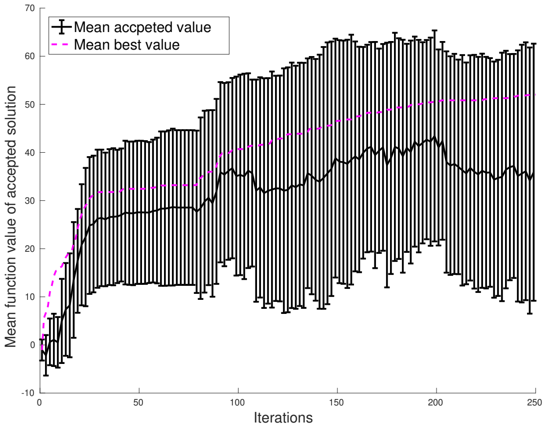
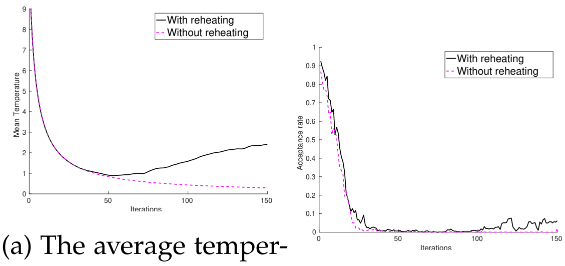


Figure 4.16: The average accepted objective function value and the average best objective function value of 24 runs with 250 iteration and 50 moves with reheating. The standard deviation of the average accepted value is shown.



(a) The average temperature over the iterations for the processes with and without reheating.

(b) The average acceptance rate for the processes with and without reheating.

Figure 4.17: The average temperature and acceptance rate for the processes with and without reheating.

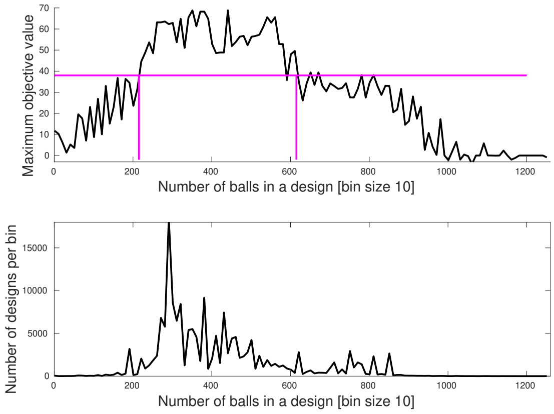


FIGURE 4.18: Average objective function value as a function of the design size. Designs with between 300 to 610 balls have, on average, better performance than the smaller and bigger designs.

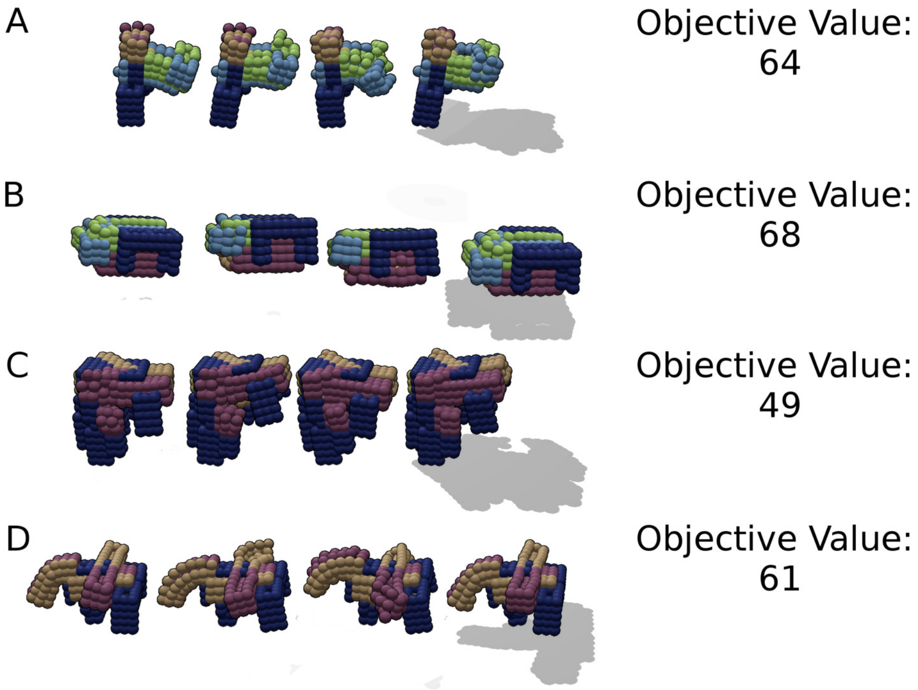


FIGURE 4.19: Designs showing hopping locomotion. Design A uses the impulse of its arms to lift its base of the ground. Design B hops by lifting off and rotating before landing. The locomotion of design C is comparable to design A. Design D pushes off with its only leg and balances using a tail-like structure.

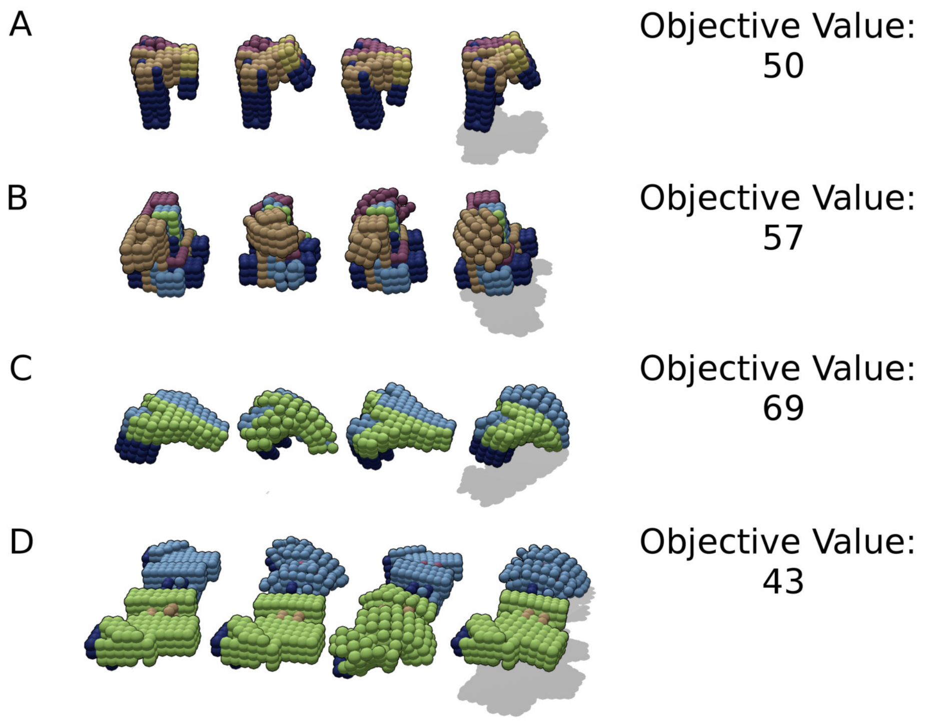


FIGURE 4.20: Walking designs. Although the propulsion of design A is very similar to that of hopping designs A and C, here, the legs are lifted off the ground alternating, resulting in steps. Designs B, C and D show legged locomotion where the leg is lifted of the floor just enough to reduce the friction.

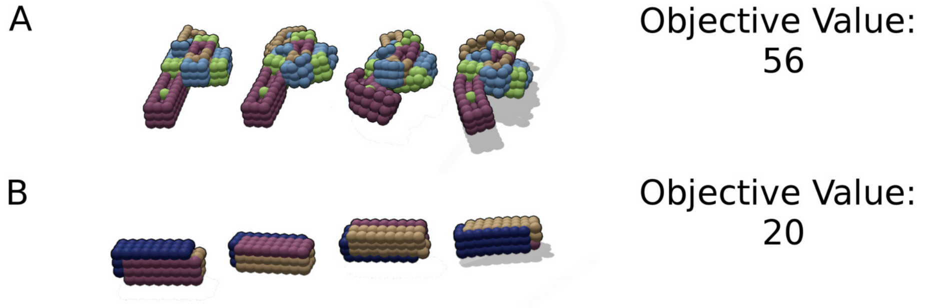


FIGURE 4.21: A crawling and a rolling design. Design A crawls and has minimal lift-off of the ground. Design B rolls.

4.3.4 *Grammar Performance*

The performance of the spatial grammar is measured by the same criteria as the grammar of study 1, see Section 4.2.3.

4.3.4.1 *The predictability of short-term rule application quality*

As can be seen from Figure 4.22, the short-term effects of the rule application of this grammar are similar to that of the grammar in study 1. On average, the rules have a negative effect, occasionally a rule application improves the design though.

4.3.4.2 *The predictability of long-term rule application quality*

Table 4.5 shows the occurrence of rules in the history of final designs. The differences between the rules is minimal. Figure 4.22 suggested this already but here it is confirmed that no rule has a special long-term effect. Table 4.5 shows that no one rule is accepted significantly more often than others, with the exception of the undo-rule.

4.3.4.3 *The anticipated behavior of sub-assemblies*

The behavior of the different assemblies in the designs is assessed. The extension of Rule 1 is found as anticipated in several final designs, e.g. Figure 4.20 B and Figure 4.21 A. The gate assembly of Rule 4 was intended to raise a design off the ground and is found in Design A in Figure 4.19 and Design A in Figure 4.20. Other designs, e.g. B and D in Figure 4.19, use the gate assembly as a counterweight. The extension in the Y-direction of Rule 2 is rarely found. The use of the leg from Rule 5 is demonstrated by design C in Figure 4.20. Variations on this V-shaped design occur very frequently in the set of final designs. A diagonal like configuration does occur occasionally in the set of final results but is relatively inefficient. The extension in the x-direction, Rule 3, is recognizable in almost all designs shown here. It enables rolling (4.21B), it generates the impulse for hopping (4.19A, C), it moves legs (4.20D) and functions as tail (4.19D).

As discussed in Section 1.1, balance is sought between design freedom and control over the design generation. In the case of the grammar presented here the question is to what degree the assemblies of the rules should be used as intended only. In an attempt to answer that for this specific grammar, a design entropy metric is defined. The entropy of a design is determined by calculating the average of the local diversity within a design. For every ball in the design, the local diversity is defined by the fraction of identical balls in its direct neighborhood. That is, a design only consisting of balls with identical parameters has an entropy value of 1.0 and a design that has no homogeneous areas of balls anywhere will have a value of 0.0. Figure 4.23 shows the relation between design entropy and the objective function value for this grammar. This means that high entropy, i.e. a homogeneous design, corresponds with a low objective function value.

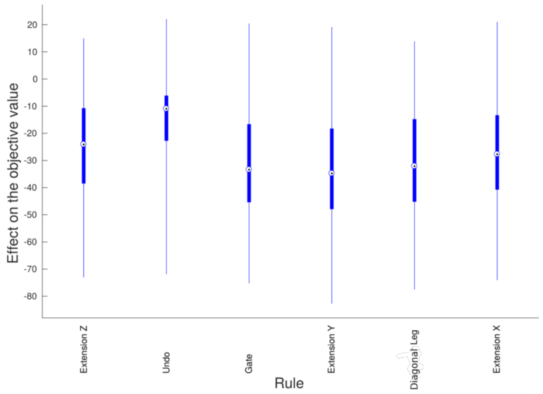


FIGURE 4.22: Average improvement of the objective function value for the different rules. All rules have, on average, a negative effect on the objective function value. A small percentage of rule applications improves the designs. The median is shown as a dot, the standard deviation as a bar.

Rule	Occurrence %
Extension z	9.99
Undo	56.28
Gate	8.62
Extension y	7.28
Diagonal Leg	8.59
Extension x	9.23

TABLE 4.5: Occurrence of rules in the history of final designs. This data is generated by recording accepted rule applications only. Note that when designs exceed the maximum size, the undo-rule is chosen.

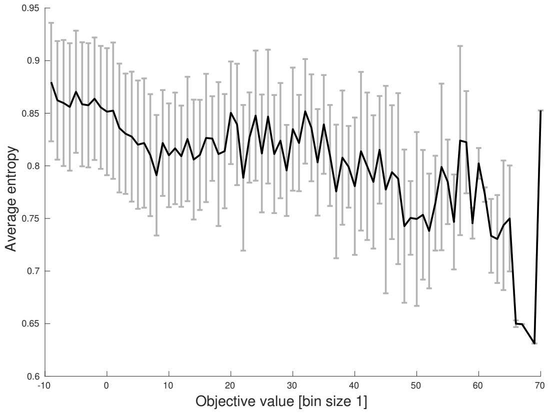


FIGURE 4.23: The design entropy versus the average objective function value. The design entropy is defined as the average of the local diversity within a design. For every ball in the design, the local diversity is defined by the fraction of identical neighboring balls.

4.3.5 *Discussion Study 2*

The reheating scheme implemented in study 2 allows the design process to keep finding better designs. Two additional parameters are added to the optimization method. With incorrect values for these parameters, convergence is difficult to achieve. However, the best design seen is saved and constant effort to improve on that design is more important than convergence.

Comparing the second grammar to the first, observing the resulting designs shows that the second grammar generates designs with less separate cluster of material and without the thin remnants of removed appendages that can be found on the design in Figures 4.8, 4.9 and 4.10. For the ease of prototyping this is an improvement. The average best objective function value of the runs of study 2 are slightly better than study 1. However, the designs without reheating are only marginally better than the designs from study 1.

The entropy versus objective function value results, shown in Figure 4.23, suggest that the selection of a grammar rule to apply could be guided toward low entropy designs, i.e. designs with a higher variety of balls.

Restricting the size of the designs proves to be justified, from a resource perspective as larger designs use exponentially more evaluation time, and from a design perspective since for larger designs have a lower average objective function value.

4.4 SUMMARY

This chapter shows that the presented spatial grammars can guide the generation of the design towards locomotive designs with appendages. The grammars limit the size and the number of separate clusters of actuated material, thus allowing fabrication with a limited number of actuators. Nevertheless, a large variety of designs are generated, showing different gaits of walking, hopping and crawling. These gaits are not explicitly programmed and emerge from the process.

Although the generation method causes the objective value function to be discrete due to the addition and deletion of clusters of material, simulated annealing is shown to find a good design in many cases.

5

COMPUTATIONAL DESIGN SYNTHESIS OF MORPHOLOGY AND ACTUATION OF SOFT ROBOTS FOR LOCOMOTION

The CDS method presented in the previous chapter consists of three phases that are repeated each design iteration (Figure 5.1a); 1) a morphology is generated, 2) the morphology is evaluated in simulation and 3) an optimization algorithm accepts or rejects this morphology. During evaluation, each morphology is actuated with a predefined actuation pattern, describing the target actuation. As a result of the fixed actuation patterns, morphologies that conform to these given actuation patterns are rewarded, i.e. the fixed actuation patterns can be seen as an optimization constraint. By using fixed actuation for the evaluation of designs, morphologies that can succeed with different actuation, but not with this one, are unsuccessful and thus the solution space is greatly decreased. Additionally, it is difficult to assess the quality of the used fixed actuation patterns, i.e it is difficult to reason what part of the morphology solution space is enabled and what part is excluded by the actuation with these patterns. To determine the real value of a morphology, it is to be actuated such that it performs according to its potential. Therefore, the simplified design task of the previous chapter is extended with an actuation pattern optimization phase.

In the design method presented in the chapter, a set of actuation patterns is optimized for each new design candidate. As shown in Figure 5.1b, the morphology design loop is extended with a nested actuation design loop. For every morphology generated, multiple actuation optimization iterations are executed. Morphologies from consecutive design iterations may differ greatly and even have a different number of actuators.

This chapter explores two possible approaches to the optimization of the actuation patterns and integrates the most suitable into the CDS. Research question 2.1, 'How can an actuation pattern be optimized for an arbitrary design within a limited time frame?', is answered in Section 5.2, where the two methods for the optimization of the actuation of soft robots are compared.

The two selected approaches for optimization of the actuation of individual designs are Parametric Actuation Curves (PAC) optimization and Reinforcement Learning (RL). PACs are a natural extension of the fixed actuation patterns and build on the assumption that the design task requires cyclic actuation. Instead of a fixed pattern assigned during generation, each actuator of a design is actuated with an actuation pattern with several variables. These variables determine the shape and the frequency of the pattern and are subjected to optimization. Since PACs are feedforward controllers, i.e. they do not take any (sensor) input, they are sensitive to small model changes. The second approach, RL, treats the problem as an unsupervised learning problem and optimizes feedback controllers. This approach enables a wider range of actuation since it is not limited to cyclic actuation. The feedback controllers resulting from RL are more robust and therefore suitable for the translation to physical robots. Additionally, closing the reality gap is an active field of research in RL. Here, a model-free RL method is considered because it is infeasible to construct a state-transition-model for the system dynamics based on the simulation models proposed in Chapter 3. To make the distinction between feedback and feedforward control clear throughout this work, PACs will be referred to as actuation patterns rather than controllers.

Based on the results of the conducted experiments with the two methods, PAC optimization is selected as the actuation optimization method. In Section 5.3, the optimization of PACs is incorporated in the design process to answer research question 2.2, 'How can the search for an actuation pattern for every design candidate be integrated in the CDS method?'.

To assess that the resulting CDS method for morphology and actuation can generate concepts that can be realized, a state-of-the-art soft robot is generated and an actuation pattern is optimized for the design. By comparing the simulated performance with the reported real performance, a first indication is given on the effectiveness of the CDS method. In Chapter 6, design resulting from the CDS method are prototyped.

This chapter is organized as follows. In the first section, the CDS of morphologies is revisited to introduce the spatial grammar used for the design generation in this chapter. The designs resulting from generation with the new grammar and fixed actuation are used to test the two actuation optimization methods in the subsequent sections. Section 5.3 shows the integration of the most suitable method into the CDS method and discusses how this improves the results. A state-of-the-art soft robot is replicated in Section 5.4.

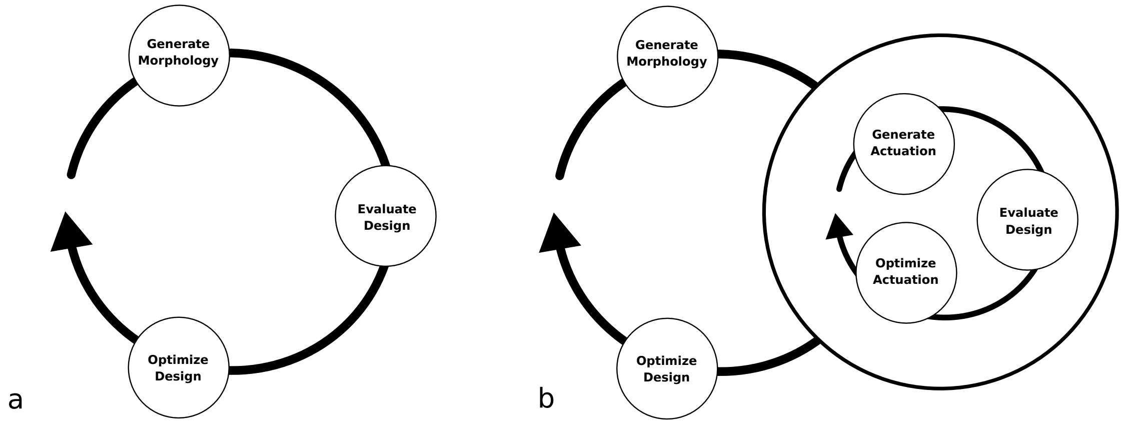


FIGURE 5.1: An overview of CDS with predefined actuation, as presented in Chapter 4 and CDS with actuation optimization.

5.1 CDS WITH CONTROL AS A CONSTRAINT

Here, the morphology generation, evaluation and optimization are briefly revisited. The approach is the same as in the previous chapters; the designs are generated using a spatial grammar, evaluated in simulation and optimized using simulated annealing. However, the two following changes are made. First, designs are simulated with the rigid-body-hinge model presented in Section 3.3.2. This model is computationally less expensive and is expected to approximate reality more accurately than the ball-spring model used in the previous chapter. However, the ball-spring model can simulate extending actuators, which the rigid-body-hinge model used here cannot. At the cost of this versatility, the rigid-body-hinge model allows for more direct model modification to approximate specific bending actuators. Second, a new spatial grammar is formulated that, similar to the new simulation model, exploits the assumption of the use of bending actuators more than the grammars presented in the previous chapter. Additionally, the new rules synthesize morphologies that correspond to the rigid-body-hinge model.

Designs are generated from building blocks, as shown in Figure 5.2a. Spheres and cubes are unactuated blocks, cylinders belong to assemblies that model soft bending actuators. The spatial grammar used for the design generation consists of four rules and is shown in Figure 5.2b.

The first rule adds a rigid part, either a cube or a sphere, at a free connection point on the design. The four parameters of the rule are 1) the geometry that is added, 2) the part of the existing design to connect to, 3) the connection point of the existing part to connect to and 4) the rotation of the newly added geometry. The second rule adds an actuator to the design. Parameters 1 and 2 are the part and connection point to connect to. The third parameter defines the length of the actuator in number of cylinders and the fourth parameter assigns one of the six fixed actuation patterns to the actuator. The six patterns are shown in Figure 5.3. The third rule modifies the rotation of a part or the actuation pattern of an actuator. The fourth rule reverts a previous additive rule application, on the condition that the reversion does not result in an invalid design, e.g. a design split in two. By mirroring each rule application in the x-y-plane, a planar symmetry is achieved. When generating symmetric designs, the mirror assembly has the phase-shifted activation of the original assembly. For example, if an actuator on the left side is activated with activation 1 from Figure 5.3, then the right will be activated with activation 4.

As in the previous chapters, the objective is to maximize the traveled distance within a fixed time and the performance of a design is assessed in simulation. The same objective value function as in Chapter 4 is used; see Equation 4.3. This weighted sum maximizes the distance traveled while minimizing the average lost potential energy.

Several design resulting from the generative design described above are presented in Figure 5.4. Design A is a result with two actuators. After a simulation period of 20 seconds, Design A has an objective function value f of 80. As can be seen from Designs B, C and D, the grammar allows for the generation of an array of distinctive, yet successful, designs.

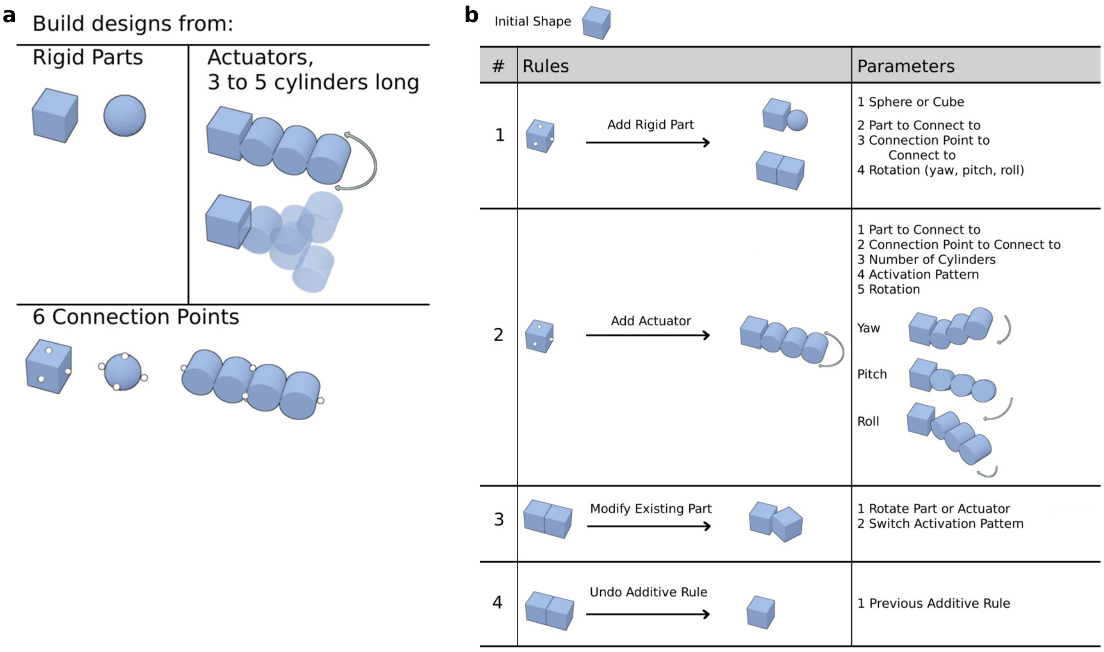


FIGURE 5.2: a) The building blocks of the spatial grammar. Spheres and cubes are single elements. Cylinders are part of bending sub-assemblies. Every building block has 6 locations to connect to other building blocks. b) The spatial grammar rules used for the design generation.

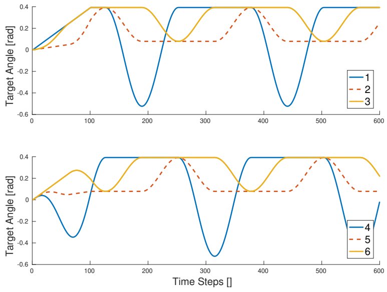


FIGURE 5.3: Six activation patterns. The bottom three are phase-shifted duplicates of the top three and are used to generate an alternating gait in symmetric designs by assigning phase-shifted patterns to the mirrored actuators.

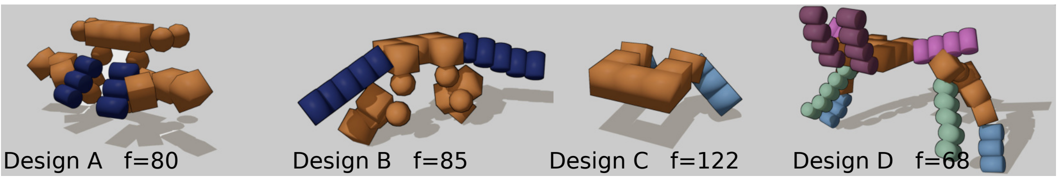


FIGURE 5.4: These designs, resulting from the generation with fixed actuation, are used in all control optimization experiments of Section 5.2. Design D has eight actuators, the other designs have two actuators. The objective function value f after 20 seconds of simulation with fixed actuation patterns are given as well. Orange balls and cubes are unactuated, stiff material. The rows of cylinders forming actuators are shown in different colors. Actuators that are symmetric counterparts are shown in the same color.

5.2 FINDING A SUITABLE ACTUATION OPTIMIZATION METHOD

To evaluate the suitability of PAC optimization and RL for the optimization of robot actuation within a CDS method, they are assessed on three criteria; 1) the performance of the designs with optimized actuation, 2) the computational costs of the method and 3) the tuning requirements of the method.

Since the rigid-body-hinge simulation model is used in the process of optimizing actuation, morphologies are needed for which it is known a successful actuation with this model exist, such that unsuccessful actuation optimization cannot be attributed to model inaccuracies. Therefore, several designs resulting from the CDS method of morphologies only are selected. An additional motivation to use generated designs as benchmarks is that this allows for the comparison of the performance of the morphology with optimized actuation to the performance of the morphology without optimized actuation.

To compare the performance and computational costs, the actuation optimization methods are used to find a controller for Design A in Figure 5.4, a design with two actuators. Whether a method needs parameter tuning for each new designs is investigated by optimizing actuation for Designs B and C, which also have two actuators. It is expected that a set of learning or optimization parameters are valid for designs with the same number of actuators, and possibly for designs with a different number of actuators. Optimizing a controller for Design D, which has eight actuators, is used to evaluate if a method performs well for designs with more actuators.

Section 5.2.1 shows the implementation and results for PAC optimization and Section 5.2.2 presents the result of using Soft Actor-Critic (SAC), a state-of-the-art RL method, for the learning of controllers.

Although SAC use a specific reward function for learning, the presented objective function values are determined according to Equation 4.3, as for fixed actuation and PAC optimization. The relevant cost measures for PAC optimization and SAC are different. For SAC results, plots and tables will show state transitions, i.e. the number of times an experience is obtained from the environment and a new action is chosen. PACs are optimized using simulated annealing. Thus for PAC optimization, the cost measure is the number of moves, i.e. the number of times the parameters are modified and evaluated. The measures are related through the simulation time and step size and can be converted into another for comparison. Subsection 5.2.3 compares the results of the two actuation optimization methods.

5.2.1 *Parametric Actuation Curve Optimization*

PACs are used to describe the actuator target angles, similar to the fixed actuation patterns, and are a form of feedforward control. One PAC describes the angle of one actuator. For symmetric morphologies, each mirrored pair of actuators has an independent actuation curve. This assumption reduces the number of PACs to be optimized for symmetric designs by a factor of two.

PACs are formulated as follows. Bezier curves are introduced to implement the PACs; see Figure 5.5. Each Bezier curve has five variables subjected to optimization, four of which define the values of 15 curve points for an order 15 Bezier curve. The fifth variable defines the mapping of the Bezier curve to the timeline, thus it determines the frequency. The start and end points of the curve are set equal to enable smooth cyclic behavior. By forming sets of three identical curve points, the curves follow the curve points closer and the first and second derivatives at start and end are matched. The control points of the curves are bound to the $[-2.5, 2.5]$ interval. The period of the Bezier curves is varied within the $[3.5, 7.0]$ interval. Every PAC has two additional binary variables. The first variable inverts the Bezier curve. For the actuation this is equivalent to inverting the bending from positive to negative. The second variable defines if a mirrored actuator is actuated in phase or in counter-phase with the original actuator. This variable is only relevant for symmetric designs.

The frequencies of a set of PACs for one design is kept equal. This modification ensures that the gait simulated for 20 seconds is representative for the gait after these 20 seconds. When the actuation frequencies of the different actuators are not equal, the frequency of the combined actuation might be much longer than 20 seconds. With that assumption, the set of variables to optimize for the PAC actuation consists of four Bezier curve points, a binary inversion value and a binary mirror-phase value per PAC and one frequency value for the set of PACs. The PACs are optimized with simulated annealing and evaluated with Equation 4.3. Every simulated annealing move, each Bezier curve point is perturbed with a probability of 0.5 and the period of the entire set of PACs is perturbed with a probability of 0.3. The size of the perturbation is defined by a normal distribution with $\mu = 0.0$ and $\sigma = \pi * T$, where T is the current annealing temperature. The binary inversion and mirror-phase values for each PAC are inverted with a probability of 0.1. The starting annealing temperature is 9.0 and is decreased every iteration according to a Cauchy cooling schedule with a cooling rate of 0.1. Reheating is used if the process has not accepted a new solution in the last 80 moves.

PAC optimization is applied to Design A and Design B, both with two actuators, and Design D with eight actuators. The simulated annealing process runs 200 iterations, with 50 moves per iteration. In Figure 5.6, the average accepted objective function value versus moves of 22 runs of PAC optimization for the three different designs is shown. Since this optimization adjusts parameters every move, and not every state transition, moves are used in the convergence plot. In Table 5.1, the PAC optimization results are compared to the performance of the fixed actuation results. Although not every run finds a better actuation, all runs find actuation patterns that reach at least 70% of the objective function value of fixed actuation, even for designs with eight actuators. For each design, multiple actuation patterns are found that outperform the fixed actuation.

5.2.2 *Reinforcement Learning*

SAC [71] is an off-policy actor-critic method with a continuous action space. Since SAC is a Deep Reinforcement Learning (DRL) method, it uses Neural Networks (NN) to approximate

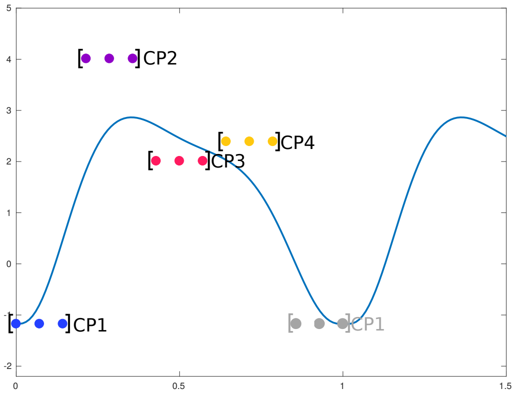


FIGURE 5.5: One and a half cycles of an order 15 Bezier curve. Every set of 3 Bezier Curve Points (CP) is aggregated and represented by a single variable for optimization. The last set is always equal to the first to ensure smooth cyclic behavior. By using an order 15 curve, the curves follow the curve points closer and the first and second derivatives at start and end are matched.

TABLE 5.1: The PAC optimization results compared to the fixed actuation result for three designs.

Design	Fixed Actuation Result	Best Result / Fixed Actuation Result	Worst Result / Fixed Actuation Result	Runs Better than Fixed / All Runs
Design A	80.0	1.16	0.99	19/22
Design B	85.4	1.26	1.22	22/22
Design D	67.6	1.20	0.73	9/22

the state-value and action-value functions in the actor and the critic. The variables subjected to learning are the weights of these networks. A schematic of actor-critic RL is shown in Figure 5.7.

Here, the environment is a soft robot in simulation. The action selected by the actor consists of a target angle adjustment for each actuator in the range $[-0.3, 0.3]rad$. To allow the selected action to have a measurable effect on the robot, a new action is selected only three times per second.

From the environment, a state and a reward are fed back to the RL method. Here, the state consists of four components illustrated in Figure 5.8. The first state component is the current actuator shape that is determined by taking the average of the angles of the hinges of the actuator. This approximation assumes a constant curvature over the actuator. The second component is the actuator speed and is measured as the difference between the current and previous actuator angle. Binary floor-contact sensors for each building block and three rotations for the center building block (initial shape as shown in Figure 5.2) form the last two components of the input.

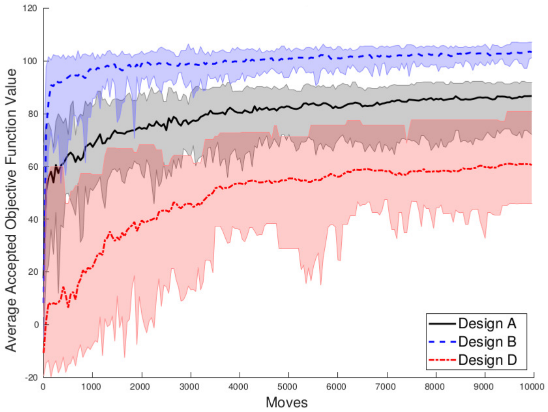


FIGURE 5.6: Average accepted objective function value, and the range, versus moves for PAC optimization. A PAC is optimized for Design A, Design B and Design D. For each design, the average accepted objective function value and the range of 22 runs is shown.

A reward function defines the reward that is used to update the critic. Following the reward structure used for a similar tasks by [71], the dense reward function is given by:

$$r_t = W_{dist} \min_i \mathbf{x}_t^i + W_{velo} \dot{\mathbf{x}}_t^c \qquad i = 1, \dots, N \tag{5.1}$$

Here, r_t is the reward at control step t, W_{dist} and W_{velo} are the weight factors for the distance and velocity components, N is the number of elements in the design, $\mathbf{x}$ is the horizontal location of element i and $\dot{\mathbf{x}}_t^c$ is the horizontal velocity of the center element, i.e. the initial shape depicted in Figure 5.2, which is present in every designs. The distance component of the reward depends on the shortest distance traversed by any element of the design, similar to the objective value function given by Equation 4.3. The distance and the velocity weights are empirically chosen to be $W_{dist} = 2.0$ and $W_{velo} = 20.0$. SAC uses a modified reward function that consists of the reward itself, as described above, and a term to maximize the entropy of the policy. A policy with high entropy is a policy small differences between the action selection probabilities, i.e. a flat Gaussian. Learning a maximum entropy policy increases exploration and a policy that is able to select good actions without unjust converging on a single action makes the method less brittle [71].

The evaluation of the policy formulated in the critic is updated and the critic is used to improve the actor. With the new state as input, the improved actor selects the next action.

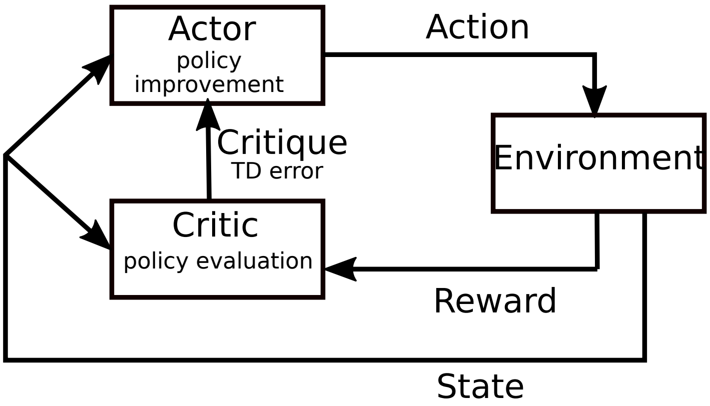


FIGURE 5.7: A schematic of actor-critic RL.

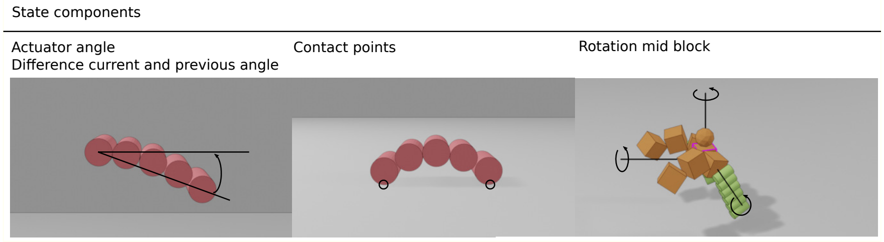


FIGURE 5.8: The four components of the input state; the actuator angle, the difference between the current and previous angle, floor-contact points and three rotations for the center building block.

To investigate the use of SAC for the control of generated morphologies, the SAC-implementation by Tuomas Haarnoja [86] is combined with the previously described simulation method.

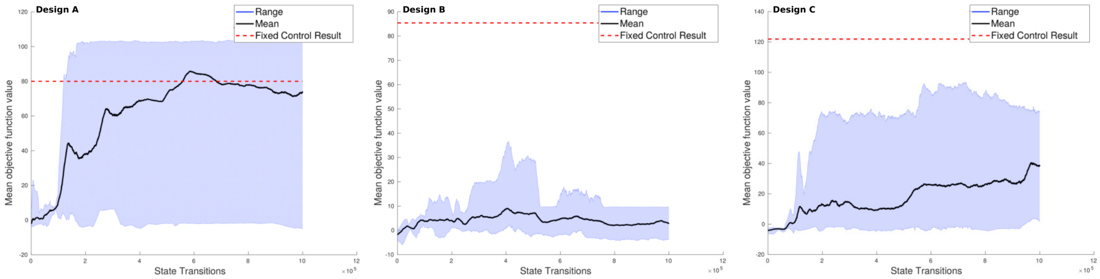


FIGURE 5.9: Average objective function value plotted against state transitions for 9 SAC runs for Design A, Design B and Design C. For each designs, the objective function value achieved with fixed actuation is plotted as red dashed line.

First, a controller for Design A is optimized. Running nine SAC experiments each with $10e5$ state transitions results in a best learned policy with an objective function value of 109.4, which is around 37% better than the fixed actuation. Figure 5.9 (left plot) shows the average objective function value and its range plotted against the number of state transitions. As can be seen, the range of the objective function values of the policies is very large. This means that not every SAC run converges to a better solution. An extensive parameter search has been conducted, nonetheless, the faltering convergence might be solved by further tuning of the reward function and learning parameters. Even though convergence is desirable, storing the best found controller is sufficient here.

When adding SAC to the CDS method shown in Figure 5.1, the morphology to optimize a controller for changes every design iteration. As a consequence, the method that optimizes the actuation has to perform without design specific tuning, or it has to be feasible to auto-tune the learning parameters and reward function. The learning parameters and reward function as used for Design A are tested for two other designs with two actuators, Design B and Design C shown in Figure 5.4. As can be seen from the middle and right plots in Figure 5.9, this does not result in good controllers. None of the 18 runs reported a best objective function value that was better than the fixed actuation objective function values, meaning that the used parameters and reward function do not fit other designs with the same degree of freedom as Design A. The results of the SAC experiments are summarized in Table 5.2.

From Figure 5.10, it is seen that the learned policy for Design A results in cyclic and symmetric behavior, although no such assumptions are made.

Given the experiments for designs with two actuators show that SAC needs design specific tuning to succeed, the method is not considered for the evaluation of designs within CDS.

Table 5.2: The SAC results compared to the fixed actuation result for three designs.

Design	Fixed Actuation Result	Best Result / Fixed Actuation Result	Worst Result / Fixed Actuation Result	Runs Better than Fixed / All Runs
Design A	80.0	1.37	0.99	8/9
Design B	85.4	0.92	0.28	0/9
Design C	121.9	0.9	0.35	0/9

A second RL method, Double Deep Q-Networks (DQN) is implemented and investigated as well. However, the method uses a discrete action space and therefore it fails for designs with more than 3 actuators. For details on the implementation and results, the reader is referred to Appendix B.

5.2.3 *Comparison of the Methods*

The two methods are compared on three metrics; 1) the maximum performance, 2) the computational cost and 3) the need for design specific tuning; see Table 5.3. For the SAC, the number of state transitions is a meaningful measure of the computational cost. For simulated annealing, however, the number of objective value function evaluations indicates the computational cost. These two measures are related through the simulation length and timestep.

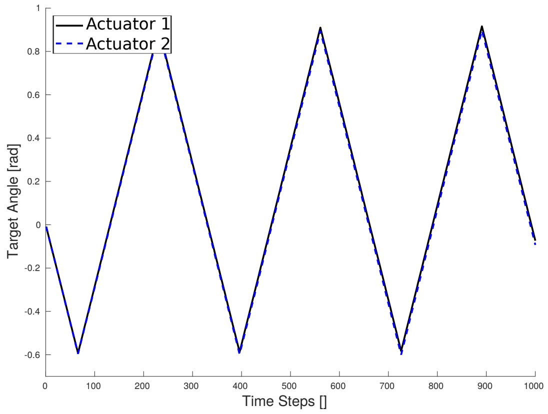


Figure 5.10: Target angles for the 2 actuators of Design A given by the policy learned with SAC. As can be seen, the actions of the 2 actuators only differ slightly and the control is cyclic.

Although SAC achieves the highest objective function value for Design A, the reward function and learning parameters used for Design A fail when used for Designs B and C. Tuning is required for every new morphology generated and would have to be automated. Further, automated tuning increases the computational cost. In contrast, PAC optimization needs fewer function evaluations and works without design specific tuning, i.e. after the initial tuning of the optimization parameters, these work for all designs.

With the goal of expanding the design space by evaluating morphologies with custom actuation, PAC optimization is selected for the co-design of morphology and actuation.

Table 5.3: Comparing the performance of PAC optimization and SAC for finding the actuation of Design A. The relevant measures for computational cost differ between the methods but are related through simulation length and time step. The converted measure, related through the simulation time and step size, is given in parentheses.

Method	Best Objective Function Value	Number of State Transitions	Function Evaluations	Design Specific Tuning
Fixed Actuation Result	80.0	-	-	-
SAC	109.4	10e5	(1.7e4)	yes
PAC	92.5	(6e5)	1e4	not necessary

5.3 CDS WITH PAC OPTIMIZATION

CDS is carried out with an inner optimization loop optimizing the feedforward actuation pattern for each design; see the schematic in Figure 5.11. The process starts by generating a morphology using the spatial grammar presented in Section 5.1. In the actuation optimization loop, PACs are optimized for the morphology by initializing a PAC, evaluating the combination of morphology and PACs in simulation, accepting or rejecting the PACs with simulated annealing and modifying the PACs variables to generate the next PACs. The actuation optimization loop is terminated when convergence criteria are met, or when a maximum number of iterations is reached. The objective function value of the best PAC is used as the objective function value of the combination of morphology and actuation. Finally, the combination of actuation and PAC is accepted or rejected. The next iteration starts with generating an modified morphology by applying a spatial grammar rule to the current design. To clarify, pseudo code of the method is given in Algorithm 5.12.

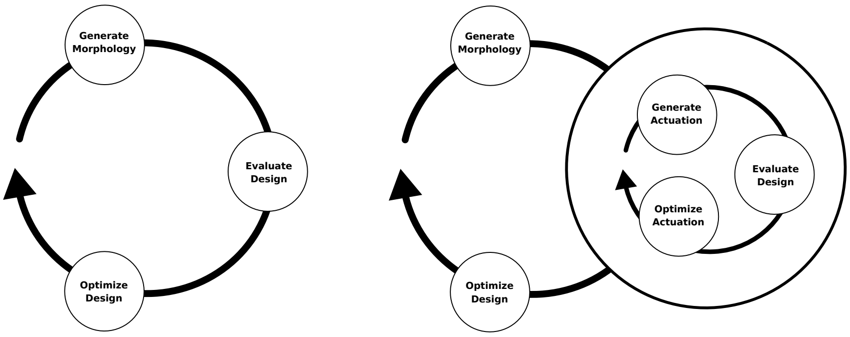


FIGURE 5.11: Schematic of CDS with an inner loop for actuation optimization.

The parameters for the actuation optimization loop, which are listed in Table 5.4, are determined empirically through optimizing the actuation pattern for Designs A-D. The parameters are a trade-off between computational costs and performance. The parameters for the outer loop, i.e. morphology and actuation combined, follow from the experiments here and the experiments conducted in Chapter 4. As can be seen from the Table 5.4, the number of moves per actuation optimization iteration is dependent on the number of actuators of the design.

By comparing the average objective function values of processes with and without actuation optimization, a potentially unfair comparison is made. Due to the modification of the actuation frequency and the amplitude of the PACs, the amount of energy a design uses is changed. Thus, designs actuated with optimized PACs potentially use more energy than designs with fixed actuation and that could be the reason for their better performance. Additionally, it is difficult to determine if the fixed actuation patterns were selected well. To make a fair comparison between a CDS with and a CDS without actuation optimization and mitigate the influence of the specific fixed actuation patterns, 30 optimized PACs from a CDS of morphology and actuation are used as fixed actuation in a design process for morphologies only. Thus, it is known that a successful morphology exist for these PACs.

Figure 5.13 shows the convergence for the three types of processes; 1) the design with fixed actuation, 2) the design with optimized actuation and 3) the design with PACs, resulting from a CDS of morphology and actuation, as fixed actuation. The figure shows that using actuation optimization improves the best objective function value by around 350% and needs fewer morphology iterations. The results of runs with optimized PACs that are used as fixed actuation confirm that these PACs are facilitate the synthesis of better solutions than the hand-designed actuation patterns used in Section 5.1. Possibly the PACs allow the designs to use more energy than the fixed actuation does or these PACs are easier to find a morphology for. In either case, finding a morphology to match a PAC is harder and less successful than evolving a morphology and PAC simultaneously.

To illustrate the variety of the actuation patterns, Figure 5.14 shows three morphologies and corresponding actuation. A diverse collection of designs that exhibit walking, rolling and jumping gaits is presented in Figure 5.15, underlining the vast solution space of the co-design of morphology and actuation presented here.

Table 5.4: Parameters used for the method given in Algorithm 5.12.

Morphology Optimization		Actuation Optimization	
T_0	9.0	T_0^c	9.0
cooling rate	0.1	cooling rate^c	0.1
maximum # of iterations	100	maximum # of iterations^c	300
# of moves	3	# of moves^c	5 for actuators<3
maximum # of parts	20		10 for 2<actuators<5
maximum # of actuators	8		15 for 4<actuators
max actuated / passive fraction	1.0	cut-off Fraction	0.05
max iterations of no improvement	5	cut-off Iters	150

Algorithm 1: Computational Design Synthesis including PAC Optimization

```
Parameters morphology optimization loop: {$T_0$, cooling rate, maximum nof iterations, nof moves,
  maximum nof parts, maximum nof actuators, max actuated / passive fraction, max iterations without
  improvement}
Parameters PAC optimization loop: {$T_0^c$, cooling rate$^c$, maximum nof iterations$^c$, nof moves$^c$,
  convergence criteria: fraction (improvement / best value) in last $\phi$ iterations is less than
  $Frac_{convergence}$ }
initialization of design $\mathcal{D}$
for maximum nof iterations do
    for nof moves do
        if maximum nof parts not reached then
            if maximum nof actuators not reached AND max actuated / passive fraction not reached
              then
                apply 1 spatial grammar (add / delete / modify) rule randomly to $\mathcal{D}$
            else
                apply 1 spatial grammar (add passive part/ delete / modify) rule randomly to $\mathcal{D}$
        else
            apply 1 spatial grammar (delete / modify) rule randomly to $\mathcal{D}$
        initialization of PAC $\mathcal{C}$
        for maximum nof iterations$^c$ do                        /* Start PAC optimization loop */
            for nof moves$^c$ do
                add $v$ to the parameters of $\mathcal{C}$, with $v \hookrightarrow \mathcal{N}(\mu = 0.0, \sigma = \pi * T^c)$
                simulate design $\mathcal{D}$ with PAC $\mathcal{C}$ for 20 seconds and determine the objective function value
                if objective function value is best then store $\mathcal{C}_{best}$;
                use SA with $T^i$ to accept or reject $\mathcal{C}$
                if convergence criteria are met then break;
            using Cauchy cooling rate$^c$, update $T^c$
        use the objective function value of $\mathcal{C}_{best}$ for design and morphology set $\mathcal{CD}$
        if objective function value is best then store $\mathcal{CD}_{best}$;
        use SA with $T$ to accept or reject $\mathcal{CD}$
    if no improvement for max iterations of no improvement then break;
    using Cauchy cooling rate, update $T$
```

FIGURE 5.12: Pseudo-code of the iterative design method.

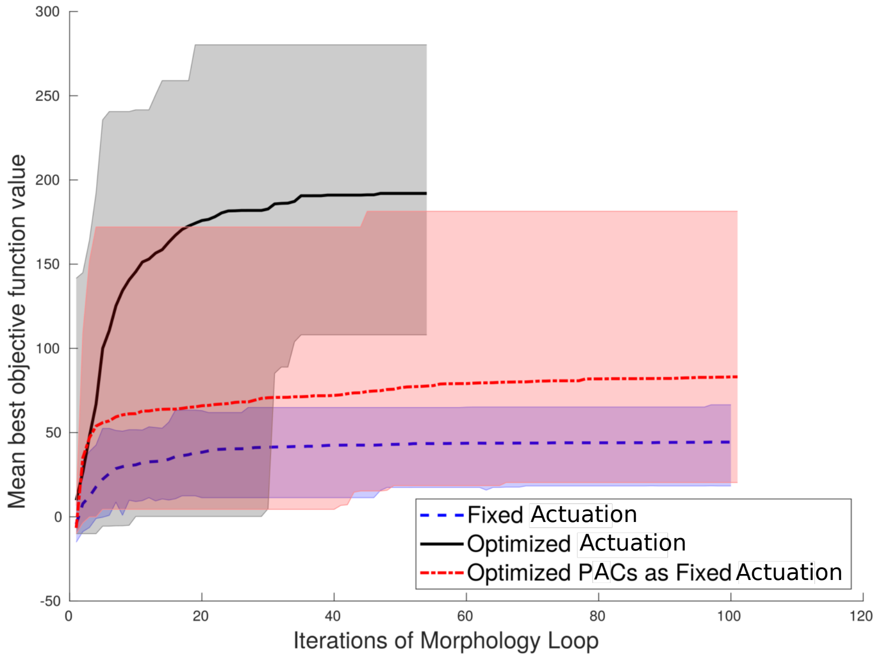


FIGURE 5.13: The average and range of the best objective function value versus the iterations is plotted to compare the design of morphology only, morphology and actuation combined and the design of morphology with PACs as fixed actuation. The design process of morphology only is run 30 times and has 100 iterations and 25 moves per run. The design process with PAC optimization is run 30 times and each run has a maximum of 100 outer loop iterations with each 3 moves and a maximum of 300 inner loop iterations with each 5 moves. The process with actuation optimization is terminated when the convergence criteria are met.

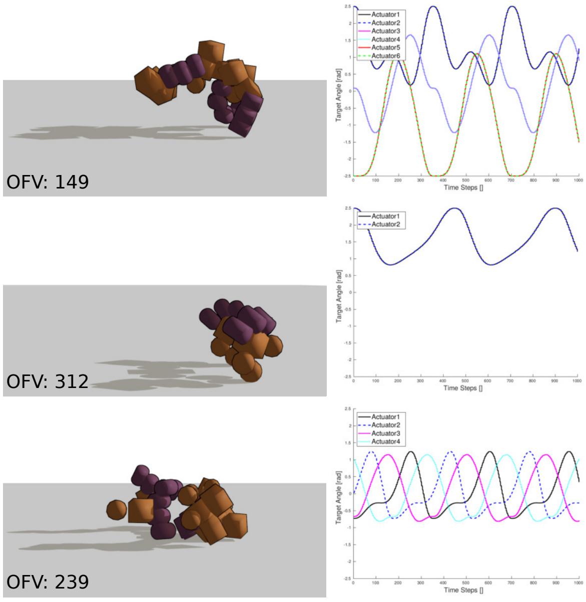


Figure 5.14: Three results from the design of morphology and control are shown. The top design has six actuators, where the symmetric actuators are actuated in phase. It uses two sets of actuators to produce a hopping motion and an extra set to keep its balance. The middle design rolls. The bottom design has four symmetric actuators in counter-phase. One pair of actuators releases the pressure on one side of the body, the other pair pushes off. For each design, the Objective Function Value (OFV) is given.

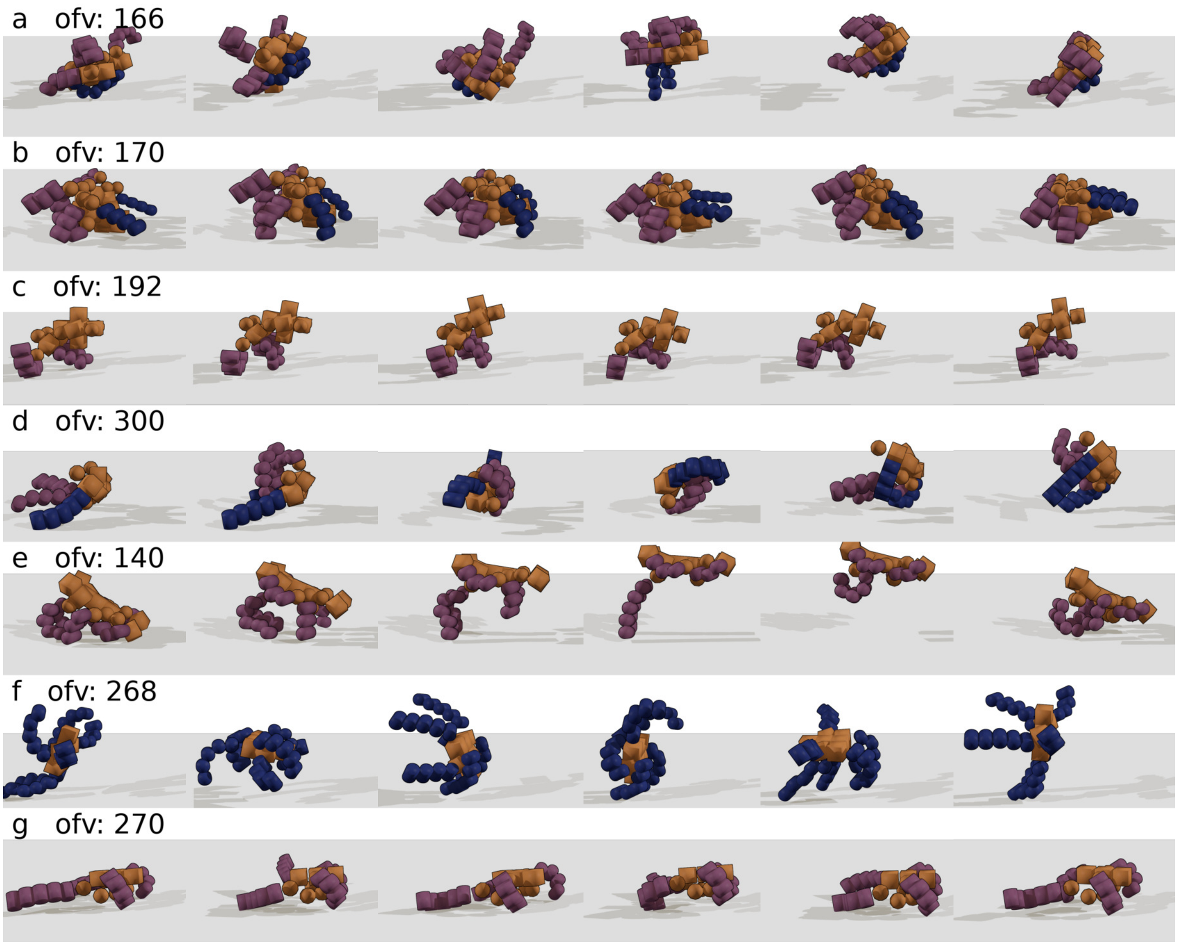


Figure 5.15: Seven designs moving from left to right. The Objective Function Value (OFV) is given on the left. Designs a and e jump, designs b, c and g walk and designs d and f roll.

5.4 GENERATING AN EXISTING DESIGN

Here, an attempt is made to generate an existing state-of-the-art soft robot to assess the capabilities of the CDS method. The robot selected is the soft robot presented by Shepherd et al. [9] and it results from research dedicated to building a soft robot for locomotion. The robot exhibits an undulating and a crawling gait. A video of the design can be found here [87]. Reverse engineering this multi-gait soft robot will answer three questions; 1) Can the generation method generate this design? 2) Does the actuation pattern optimization find the same two gaits? 3) Is the simulated performance comparable to the reported performance?

First, the morphology is replicated by hand-picking six rules from the spatial grammar; see Figure 5.16a. The dimensions of the rules are modified to equal the actuators used for this robot. Thus, it is assessed that solution space spanned by the proposed spatial grammar includes this design. As can be seen in Figure 5.16a, the physical robot has an extra actuator in the spine, reportedly to lift the center of mass and overcome friction. This actuator would have four connection points and is not possible within the used grammar rules. However, this extra actuator is needed because the molded silicone has low stiffness when unactuated and can be seen as a part of detail design, not conceptual design.

Second, the weights and forces reported in the paper are used to adjust the simulation model and PAC optimization is used to find an actuation pattern for the design. Both the undulating gait and alternating gait executed by the physical robot are found as possible gaits by the PAC optimization process; see Figure 5.16b. Shepherd et al. report velocities for both an undulating gait (13 m/h) and a crawling gait (24 m/h), expressed in body lengths per hour this is 93 and 192 body lengths per hour. The conceptual design in simulation reaches 150 body lengths per hour with an undulating gait and 291 with a crawling gait. Thus, Shepherd's prototype reaches 62% and 65% of the simulated speeds. Since the actuation in the simulation is described by a target angle and the constraints cannot be expressed in forces directly, no comparison of the actuation force or pressure can be made.

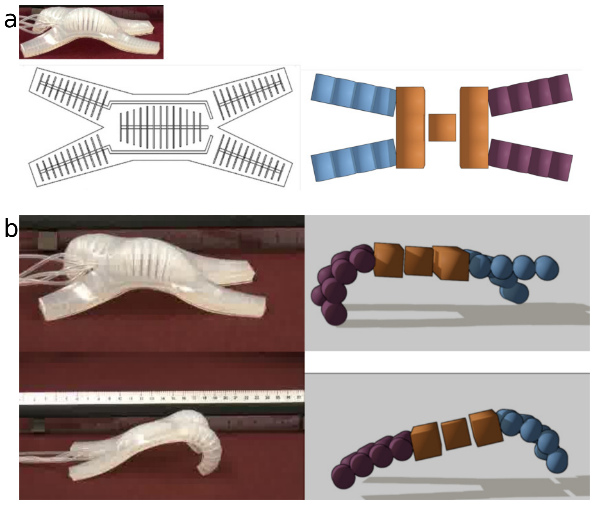


FIGURE 5.16: a) The hand-designed physical soft robot presented by Shepherd et al. and the same design generated by hand-picking grammar rules. b) The alternating gait (top) and the undulating gait (bottom) of the physical robot from [9] and the generated equivalent.

5.5 DISCUSSION

Two different approaches are investigated for the optimization of the actuation of the designs, PAC optimization that assumes cyclic actuation and SAC that is not limited to cyclic actuation.

The proposed PAC optimization method improves the performance of the designs that resulted from the design of morphology only in most runs. All but one optimization parameters work for the optimization of PACs for all designs, without modification or re-tuning. The exception is the number of simulated annealing moves for the optimization of the PACs, which is dependent on the number of actuators. However, this is adjusted automatically. The domain specific knowledge encoded in the PAC formulation, i.e. cyclic actuation and symmetry assumptions, restricts the solution space but enables the optimization of actuation for up to eight degrees of freedom in foreseeable time. For more complex tasks, it will be increasingly difficult to formulate valid assumptions to base the PAC formulation on, e.g. when a task is non-cyclic.

The experiments with SAC result in controllers with high performance. The resulting feedback controller is a good starting point for a controller for a physical robot. It is also a good base for learning more complex tasks, e.g locomotion on rough terrain or locomotion with a sense of direction. However, the computational costs are high, especially for designs with a high number of actuators. The decisive argument against using SAC as part of a CDS method is that the reward function and the learning parameters require design specific tuning. Since SAC has a relatively small number of tuning parameters for an RL method, a form of auto-tuning might be conceivable, however, this needs investigation.

The SAC results demonstrate cyclic and symmetric actuation, without these characteristics being constraints. This justifies using these assumptions for PAC optimization for a locomotion task with symmetric morphologies.

For the problem of finding an actuation pattern for an arbitrary conceptual design and with limited computational resources, an actuation generation method that limits the actuation solution space shows to be advantageous. This finding is analogues to the findings concerning the generation of soft robots morphologies. There too, the use of a spatial grammar that encodes design knowledge specific to the task has proven to be beneficial, as shown in Chapter 4.

The co-design of morphology and actuation outperforms the design of morphology in two aspects. First, fewer morphological design iterations are needed and second, the final performance is better. Additionally, the solution space is expanded since the fixed actuation patterns can be formulated as PACs and therefore, the set of all designs with fixed actuation is a subset of all designs with optimized actuation.

Finally, replicating a state-of-the-art soft robot shows that this design can be generated with the presented spatial grammar, i.e. the design is part of the solution space. Both actuation modes of the physical robot are found by the PAC optimization method. Although the actual speed of the robot is lower than that of the simulated robot, for both gaits the actual speed is around 60% of the simulated speed. This suggests that the simulation systematically overestimates the design, however, that makes the actual performance of an optimized third gait predictable.

5.6 SUMMARY

In this chapter, two methods are presented that can find an actuation pattern for an arbitrary soft robot design. A comparison is made to determine which one is the most suitable to be integrated in the CDS of soft robots for locomotion. The first method optimizes parametric control curve and results in a feedforward controller, i.e. a set of actuation patterns. This method exploits the assumption that symmetric designs require symmetric actuation and the assumption that the actuation is cyclic. The second method, soft-actor-critic, is a deep reinforcement learning method that learn a feedback controller for a design. Parametric control curve optimization improves the performance of the benchmark designs resulting from the CDS of morphology with fixed actuation and needs no parameter re-tuning between different morphology iterations. For soft-actor-critic, however, the parameters of the reward function and the learning process need re-tuning for each new design and this is not practicable within a CDS. Parametric control curve optimization is integrated in the CDS and shows to increase the objective function values by a factor of two while needing half the number of morphology design iterations when compared to the CDS of morphology only. It is shown that the presented CDS method can generate a known design and find the two reported gaits.

6

PROTOTYPING GENERATED SOFT ROBOT DESIGNS

The previous chapters show the generation, evaluation and optimization of conceptual designs of physical soft robots. In this chapter, the detail design and prototyping of several concepts is realized, finalizing the design process. The chapter answers research question 3.1; 'How can the design concepts be detailed and prototyped?' and research question 3.2 'How can insights from the process of prototyping be integrated directly in the generative design method?'.

To demonstrate the versatility of the design method, designs are detailed and prototyped with two different actuation principles. Section 6.1 presents two methods for the detail design and prototyping of the robotic concepts. The adaptations made to the CDS method to generate the two types of soft robots are discussed in Section 6.2. In section 6.3, the resulting designs are presented. The performance of the prototypes with respect to the expected performance is evaluated, as well as the performance with respect to state-of-the-art soft robot designs. Finally, the chapter is discussed in Section 6.4 and a summary is given in Section 6.5.

6.1 PROTOTYPING METHODS

The two actuation principles used are pneumatic actuation and tendon actuation. As the design generation presented in the previous chapters, the prototyping focuses on bending actuators only.

6.1.1 *Prototypes using Pneumatic Actuation*

For the first prototypes, the pneumatic actuator design from Figure 6.1 is used, whose design is loosely based on the molded silicone bender design from Dreyer [74]. The actuator results from an iterative design process and is designed in three lengths; 6cm, 8cm and 10cm. First, the process of prototyping is presented. Second, the constraints following from the fabrication are integrated in the design method.

The actuators are designed for additive manufacturing using the Form 2 from Formlabs [88], a stereolithography (SLA) machine. The material used is called Elastic and is also developed by Formlabs. The printing process consists of three stages, the printing, the cleaning and the after-curing of the printed parts. The printing, illustrated by Figure 6.2, involves a laser mask projected on the bottom of a tray filled with resin. The laser locally cures the resin and creates a single layer of the design. To detach the newly cured layer from the tray, the tray slowly slides to the right before the build-plate is lifted from the resin. This step is precarious since the part is still fragile and tears easily. Additionally, the shear introduced by this motion does not always dissolve before the next layer is printed, causing a layer mismatch. To prevent shearing and tearing, designs require internal support structures. When printing of the part is finished, the part is washed in isopropylalcohol to remove all remaining resin. For hollow parts with a single opening, it is difficult to remove the viscous resin, especially when the internal support structure prevents a free flow. In the last stage, the part is cured with UV-light at 60°Celsius.

The main challenge in inflatable actuator design with this fabrication method is finding a geometry that satisfies the following three requirements: 1) allow for a sufficiently large bending angle, 2) sustain 1.5 bar of internal pressure and 3) be printed with limited internal support. As can be seen from the cut view of a printed actuator of an earlier design iteration in Figure 6.3, the internal support and the residue resin significantly alter the internal geometry of the actuator. Where the narrow part of the original design has a smooth surface to sustain high pressure and is large enough for free flow during washing, the realized design has weak points for a tear to start and propagate and the narrow passage is almost completely blocked by internal support. To mitigate this problem, the actuators are placed on a support structure and are oriented with their openings downwards during the curing stage. The high temperature in the curing station lowers the viscosity of the residue resin and lets it drip out.

The non-actuated body parts are made of ABS plastic and designed to lock the actuators in the correct position and orientation. The ABS parts are additively manufactured using the Stratasys uPrint fused deposition modeling (FDM) machine [89]. Two reasons motivate the decision to print the non-actuated body parts separate and in ABS, as opposed to Elastic material.

First, printing the entire robot in one go, as illustrated by Figure 6.4, results in problems with the air-tightness of the actuators. As described above, the orientation of the actuators during printing and curing determines the quality and printing a complete design limits the control over the actuator orientation.

Second, Formlabs Elastic material does not bind well with any glue that was tried. Thus, the union of soft parts is realized by applying a layer of uncured resin on the contact areas and curing the assembly. However, during actuation, the pnuematic actuators burst and need replacement regularly.

To reduce the friction between the underground and the rather sticky Elastic material, a Teflon sheet is used as underground. The pressure is provided by a combination of valves that resulted from the Bachelor work of Theophile Messin-Roizard [90] and supplied by wall pressure available in the experimentation room; see Figure 6.5. The prototypes are powered by 1.5 bar air pressure. Every actuator is controlled by two valves, one valve to inflate the actuator and one to deflate the actuator. The valves in turn are controlled through an Arduino [91] micro controller. There is no sensor feedback, making the pressure control system completely feedforward.

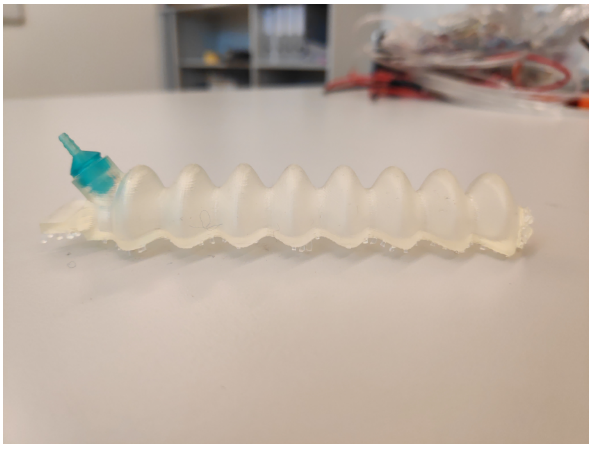


FIGURE 6.1: A pneumatic actuator realized through additive manufacturing.

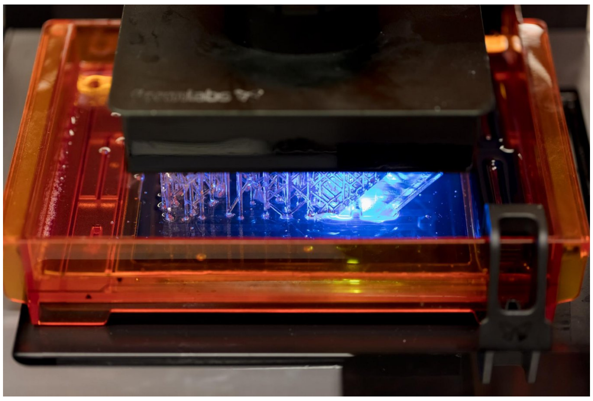


FIGURE 6.2: A step in the printing process on the From 2, image from [88]. A laser mask projected on the bottom of the tray cures a single layer of the design. To detach the newly cured layer from the tray, the tray slowly slides to the right before the build-plate is lifted from the resin.

6.1.2 *Prototypes using Tendon Actuation*

The second actuation principle used here is tendon actuation; see Figure 6.6. The actuators are printed on the Formlabs using the same process as the pneumatic actuators. A thread is inserted through the holes along one side of the actuator. By pulling on the thread, the actuator bends. Bidirectional bending is possible by threading both sides of the actuator. Alternatively, one side can be threaded with an elastic string and secured in the bended position. This way, bidirectional bending with a single control thread is possible; see Figure 6.6d and 6.6e.

In contrast to the pneumatic prototypes, the non-actuated bodies in these prototypes are printed in the same elastic material as the actuators. However, the designs are too large for the Formlabs print plate and have to be printed in parts. Parts are assembled by applying a layer of resin and curing the assembly in the curing station. Since the parts have already

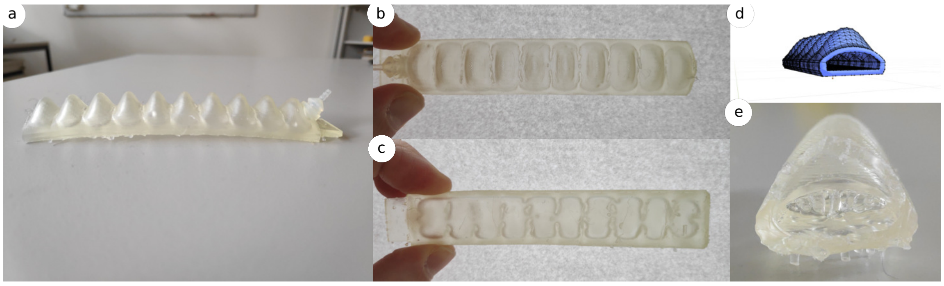


Figure 6.3: An early iteration pneumatic actuator is shown in (a), the same actuator is shown in the top view (b) and the bottom view (c). The narrow parts of the actuator are partly filled with (cured) residue resin. Figures (d) and (e) show the cross-section as designed and as printed. The cross-section is filled with support columns that make cleaning near impossible.

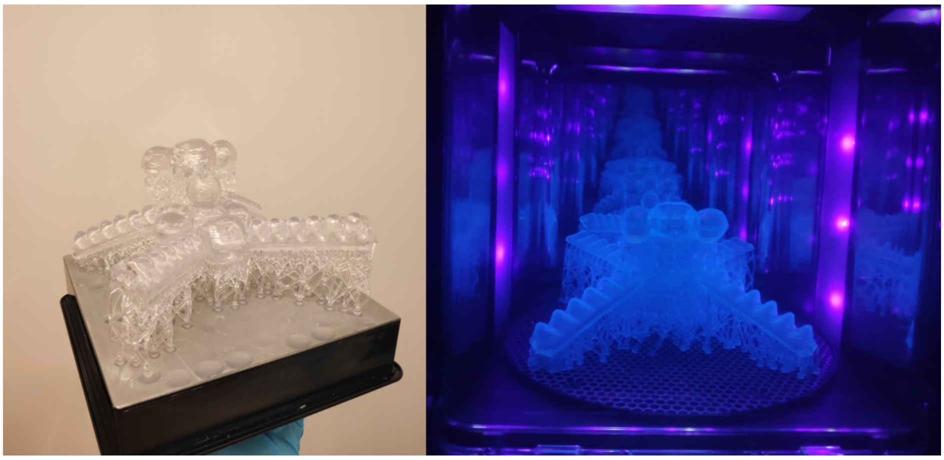


Figure 6.4: Pneumatic prototype printed in one go on the Formlabs SLA machine.

been cured once, they can be folded and compressed in the curing station, and thus, designs that are larger than the print plate size can be fabricated.

The setup to control the tendons consists of a set of six servo motors; see Figure 6.7. Since these motors operate between -90 and 90 degrees, each motor is mounted with a wheel to achieve enough angular displacement. To each wheel, a thread for positive bending and a thread for negative bending of a single actuator is connected. Ideally, the thread displacement at the wheel is equal to the thread displacement at the actuator, without exerting any force on the robot as a whole and while allowing the robot to move with respect to the wheel. Three setups are built to try to approximate this situation.

In Figure 6.7a, the motor setup is shown. Per actuator, only one thread is used and each thread has its own PVC pipe to guide it. The used thread is a soft fishing line. One of the printed joints connecting the pipes is shown in Figure 6.7b. A challenge in designing these joints is to guide the thread through the center, or neutral axis, of the joint as much as possible, without introducing additional friction. When a thread is not constrained to the neutral axis, it can take a 'shortcut' when the joint is bent. Consequently, pulling the thread exerts a bending force in the joints and the guiding pipes are no longer compliant.

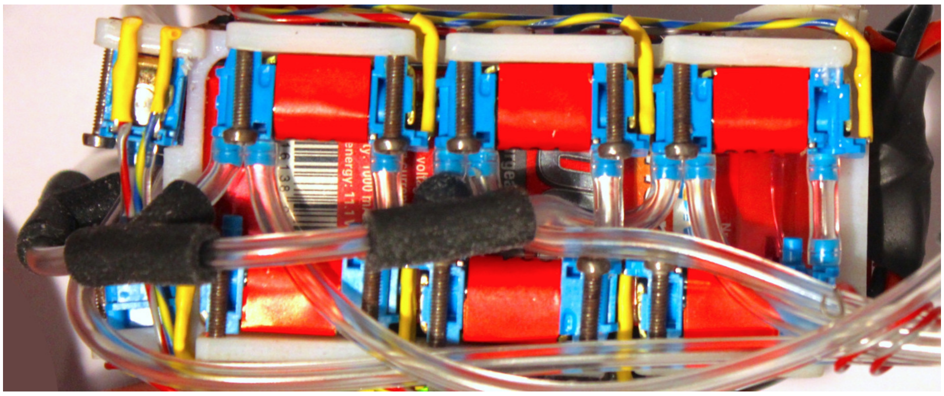


FIGURE 6.5: The system of valves to control the pneumatic actuators. Picture from [90].

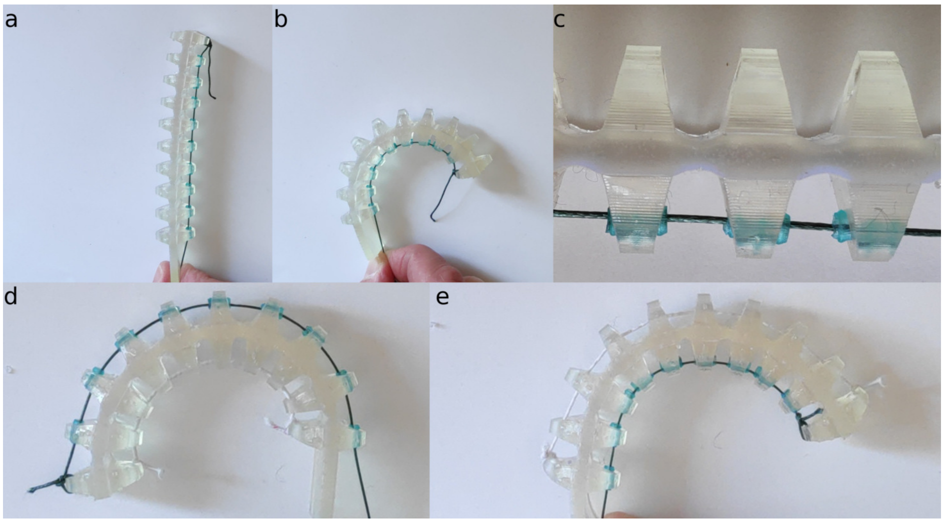


FIGURE 6.6: Tendon actuators. An actuator wired on one side in a) un-actuated position and b) actuated postion. c) A close-up showing the inserts. Pictures d) and e) show an actuator wired with an elastic thread on one side and a non-elastic thread (green) on the other.

6.2 DESIGN GENERATION

The design that are detailed and prototyped in this chapter are designed using the CDS from Chapter 5; they are generated with the spatial grammar presented in Section 5.1 and simulated for 20 seconds with the rigid-body-hinge model from Section 3.3.2. PAC optimization is used to optimize the actuation patterns and simulated annealing is used to optimize the combined morphology and actuation. The optimization parameters as given in Chapter 5 are used.

However, the objective value function used in the previous chapters, Equation 4.3, is modified. Since the evaluation is based on 20 seconds of simulation and the maximum actuation period is seven seconds, the simulation is not long enough to guarantee the performance of the design after 20 seconds. Since increasing the simulation time is accompanied by increasing computational costs, the objective value function is extended with two terms. The

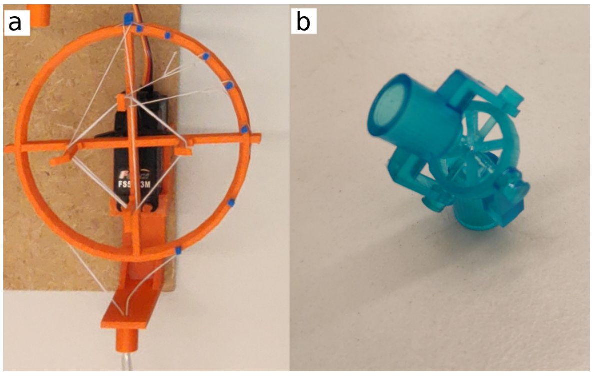


FIGURE 6.7: a) The setup of one motor as used for the actuation of the tendon-driven prototypes. To achieve enough thread displacement, the servomotor is mounted by a wheel. b) A printed joint to connect the pipes and guide the thread through the middle of the joint.

resulting objective value function to be maximized is given in Equation 6.1. The objective function value $f(\mathcal{D})$ of design $\mathcal{D}$ is given by:

$$f(\mathcal{D}) = \frac{1}{\max(P_{nc}, 1.0)} \min_i |\mathbf{x}_i^{t_0} - \mathbf{x}_i^{t_{end}}| - \max\left(\frac{1}{N}\sum_{i=1}^{N} y_i^{t_0} - y_i^{t_{end}}, 0\right) - \max(P_{nl}, 0.0) \quad (6.1)$$

where design $\mathcal{D}$ consists of N building elements. Each element i has a location $\mathbf{x}_i$ in the horizontal plane and a height y_i. Time is denoted by t. Two terms are identical to Equation 4.3 and signify the distance traveled by the building element that moved the least and the average lost height of the building elements. The two added terms are a penalty for non-cyclic motion P_{nc} and a penalty for disproportional little motion during the last third of the 20 seconds of simulation P_{nl}. They are formulated as follows.

$$P_{nc} = \frac{1}{VSN}\sum_{v=1}^{V}\sum_{s=1}^{S}\sum_{i=1}^{N} |R_i^s - R_{vi}^s| \quad (6.2)$$

$$P_{nl} = \frac{1}{3} DT^{t_{total}} - DT^{t_{last\frac{1}{3}}} \quad (6.3)$$

where R_i^s is the rotation of element i at interval s of the benchmark period of the actuation pattern. The benchmark period is the first period after an initialization time of five seconds. At ten intervals s during the benchmark period, the rotations of N elements i are stored for comparison, i.e. S is ten. During all subsequent V periods v, the difference between the original rotation of an element R_i^s and the rotation during a period v, R_{vi}^s is accumulated. The result is divided by the number of periods, the number of intervals per period and the number of building elements to obtain the penalty for non-cyclic motion. By minimizing the penalty value to 1.0, cyclic motion can not result in a reward.

A last loophole for designs to receive an undeserved high objective function value is when a design traverses a distance in the first seconds of the simulation only, but without losing height. Therefore the actual distance traveled by the middle cube of the design during the last third of the simulation, $DT^{t_{last\frac{1}{3}}}$, is subtracted from the expected distance traveled by the cube during the last third of the simulation, $\frac{1}{3}DT^{t_{total}}$.

Several modification to the spatial grammar constraints and the simulation model are made based on the selected actuation and fabrication methods. For each of the actuation methods, pneumatic actuation and tendon actuation, the following two sections present the modifications to the generation and simulation phases of the CDS.

6.2.1 *Modifications for Pneumatic Actuation*

To design specifically for the fabricated pneumatic actuators presented in the previous Section 6.1.1, the generation and simulation stages are modified. The first adaptation is to limit the bending of an actuator to positive angles, i.e. an actuator bends in one direction like human fingers do. Due to the difficulties with assembly, a second modification prevents actuators to be attached to other actuators and allows them to be attached only to non-actuated parts of the design. These prototyping constraints are integrated in the spatial grammar demonstrating the advantage of the approach to generate designs that can be realized.

Based on the measured weights of a printed ABS cube and sphere, the weights of the objects used for simulation are updated. Similarly, the weight of the simulated actuators is made to correspond to the printed actuators.

The PACs optimized for a design prescribe the angles of the actuators over time. The pneumatic system to actuate the design has no sensors and it is therefore difficult to achieve these bending angles exactly in reality. Instead, the actuation pattern that is optimized is the switching pattern of the valves, this way the actuation values optimized can directly be used in the prototype. A period of the cyclic pattern has four phases; 1) both inflation and deflation valves are closed, 2) only the inflation valve is open, 3) both are closed and 4) only the deflation valve is open. The variables subjected to optimization are the frequency and the duration of the phases.

To identify the actual actuator behavior, one actuator is inflated and deflated by switching its two valves and the inflation and deflation velocities are measured, as is the force exerted by the actuator tip at different angles.

The bending velocity of an actuator depends on if it is inflating or deflating. With the presented actuator and pneumatic setup design, the material stress causes the actuator to restore to the neutral position and this force is lower than the forces induced by the increase of pressure during inflation. The measured actuator speeds are used to adjust the increment on the bending angle per time step in simulation.

The tip force measurement cannot be translated to the simulation model directly. The hinge constraint between two cylinders belonging to an actuator has a parameter for the target velocity to achieve the desired angle and a parameter for the maximum force that can be used to reach that velocity. Together with the time step size of the simulation and the number of constraint solver iterations, these parameters define the impulse applied to the tip to resolve the collision of the actuator tip with an object. The parameters are adjusted such that this impulse corresponds to the force measurement on the real actuator.

The spring constraints between the non-actuated parts are made stiff to agree with the ABS detail design.

6.2.2 *Modifications for Tendon Actuation*

For the generation of tendon driven designs, the original formulation of PAC and bidirectional bending are used. Although actuators connected to actuators are possible with tendon actuation, for ease of prototyping the generation is restricted to actuators connected to non-actuated material. Again, the weight of the printed actuators and non-actuated components is measured and the parameters in simulation are updated accordingly.

To translate the desired angles described by the PAC to thread lengths, the printed actuators are actuated to measure the relationship between thread length and actuation angle. The bending velocity in simulation is limited to correspond to the maximum speed of the motors.

6.3 RESULTS

In Figure 6.8, the four designs that are prototyped using pneumatic actuators are presented and in Figure 6.9, the three tendon actuated prototypes are shown. Designs E and F have little sideways stability and are sensitive to small forces from the guiding pipes. Therefore, the designs have drinking straws attached to achieve enough sideways stability to counteract the interference from the pipes.

The predicted performance, the performance of the prototypes and the ratio between them are reported in Tables 6.1 and 6.2. A video showing the motion of all seven prototypes alongside their simulated counterparts can be found here: https://youtu.be/6D7EbkRiAsk.

TABLE 6.1: The predicted performance and the performance of the prototypes shown in Figure 6.8.

Design	Predicted performance (simulated)	Performance of the prototype (reality)	Ratio reality/simulated
Design A	35.6 m/h	5 m/h	0.14
Design B	75.1 m/h	6 m/h	0.08
Design C	35.6 m/h	5 m/h	0.14
Design D	56.3 m/h	6 m/h	0.11

TABLE 6.2: The predicted performance and the performance of the prototypes shown in Figure 6.9.

Design	Predicted performance (simulated)	Performance of the prototype (reality)	Ratio reality/simulated
Design E	43.4 m/h	15 m/h	0.35
Design F	137.3 m/h	28 m/h	0.20
Design G	164.3 m/h	36 m/h	0.22

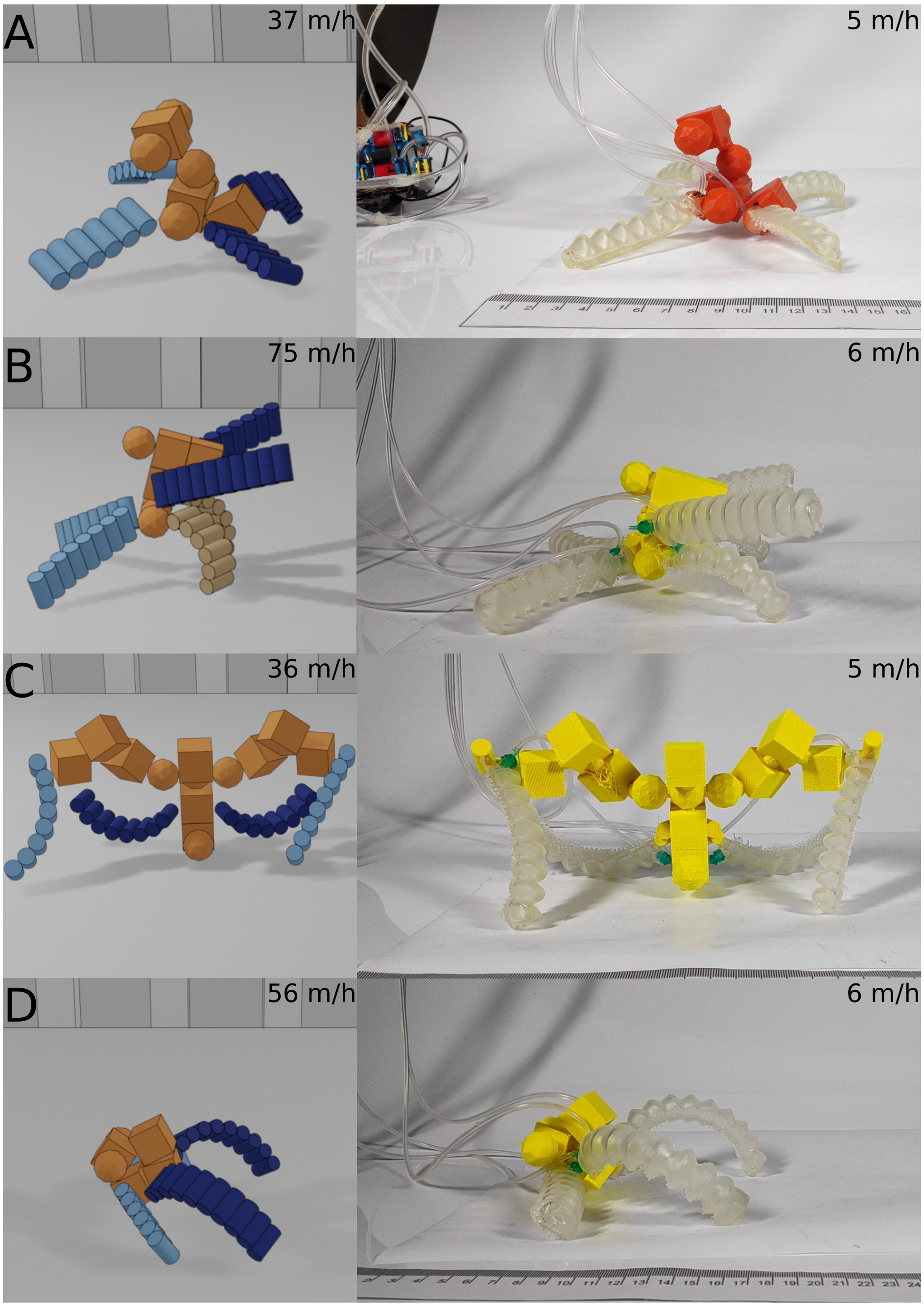


FIGURE 6.8: Four pneumatically actuated designs, simulated and prototyped.

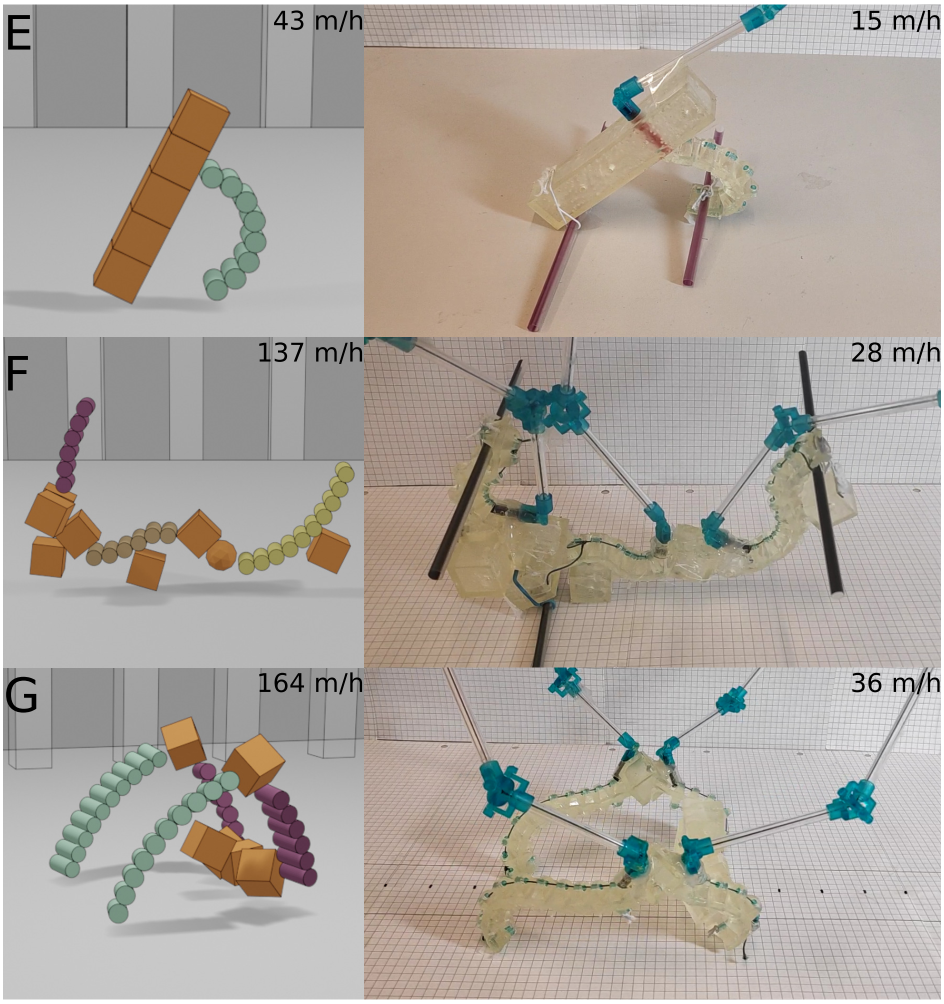


FIGURE 6.9: Three tendon actuated designs, simulated and prototyped.

6.3.1 *Comparing the Prototypes to State-of-the-Art Soft Robots*

To put the performance of the prototypes in perspective, they are compared to a collection of state-of-the-art soft robots found in literature. The first metric for comparison is the speed expressed in robot lengths. The task definition in Chapter 1 states that the task is to maximize the distance a robot traveled within a set time, thus the effectiveness of the robots is put into perspective. Since this work is partly motivated by the need for a method to generate design concepts that go beyond bio-inspired design, the second metric for comparison is novelty.

In Table 6.3, the speed of the prototypes is expressed in robot lengths to be able to compare speeds between robots with a different scales. Figure 6.10 visualizes how the prototypes perform compared to state-of-the-art soft robots.

TABLE 6.3: The predicted performance and the performance of the prototypes shown in Figure 6.8.

Design	Speed in Robot Lengths	Robot Length
Design A	39 /h	132 mm
Design B	43 /h	140 mm
Design C	22 /h	220 mm
Design D	80 /h	80 mm
Design E	150 /h	100 mm
Design F	117 /h	240 mm
Design G	240 /h	150 mm

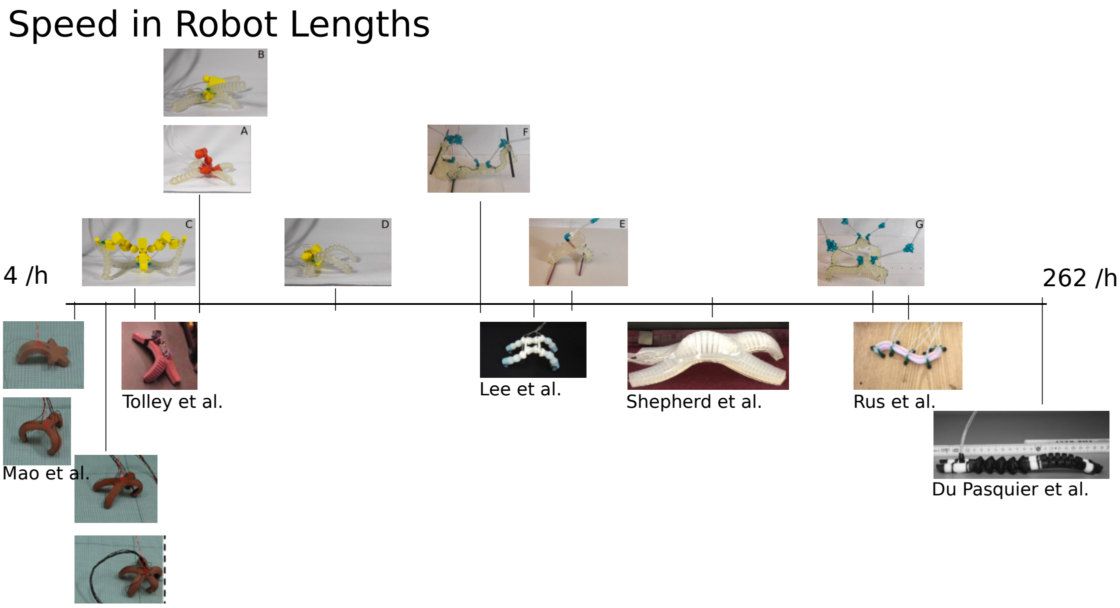


FIGURE 6.10: The speed in robot lengths of the different prototypes compared to several other soft robots. The referenced robots can be found in [21], [10], [92], [9], [22] and [39].

To quantify the novelty of a design, the novelty metric defined by Jami Shah [93] is used. He states that novelty M_1 is given by:

$$M_1 = \sum_{j=1}^{m} f_j \sum_{k=1}^{n} S_{1jk} p_k \tag{6.4}$$

Where M_1 is the novelty score for a design with m functions or features and n stages. Examples of functions are motion of a robot or method of propulsion, examples of stages are the conceptual stage or the embodiment stage. Weight f_j defines the importance of function j, weight p_k defines the importance of stage k. The specific novelty S_{1jk} can be calculated in two ways; *a priori* and *a posteriori*. In the *a priori* method, a group of experts assigns a (subjective) novelty score S_{1jk} to every idea or function thinkable. In the *a posteriori* method, an idea or function is scored by evaluating the frequency of occurrence in a collection of relevant designs. Thus, S_{1jk} is calculated from:

$$S_{1jk} = \frac{T_{jk} - C_{jk}}{T_{jk}} \cdot 10 \tag{6.5}$$

T_{jk} is the total number of ideas found to fulfill function j at stage k and C_{jk} is the count of the evaluated idea as a solution for function j at stage k.

Since the most objective method is preferred, the functions are scored with the *a posteriori* method. The functions evaluated here are: the number of actuators, the type of gait, symmetry, the maximum number of connection points of an actuator, the number of actuators per limb and if the actuators are bi- or unidirectional bending. Only the conceptual stage is evaluated, so $n = 1$. The weights for the different functions f_j are all 1.0. The collection of soft robots used to calculate the novelty of the designs presented in this thesis can be found in Appendix C.

TABLE 6.4: The novelty scores of the prototyped designs. The first row reports the average novelty of all the designs in the comparison set. The novelty scores of the individual benchmark designs can be found in Appendix C.

Design	Novelty score
Average of the benchmark designs	23.6
Design A	20.0
Design B	15.3
Design C	21.3
Design D	15.3
Design E	36.7
Design F	43.1
Design G	40.6

6.4 DISCUSSION

The prototyping presented in this chapter shows that the CDS approach presented in this thesis can be used for different actaution types and prototyping methods without major modifications.

The proposed prototyping method for pneumatically actuated robots is challenging. The SLA printing causes irregularities in the parts due to the sideways motion of the print plate that helps the part release from the tray surface. Where these irregularities cause little problems for the tendon actuated designs, the pneumatic actuators leak and sometimes burst.

The performance of pneumatic prototypes is moderate, compared to the simulated performance as well as the performance of other soft robots. The discrepancy between the simulated and the real performance is in part due to overestimation of the prototyped actuators. The actuator characterization is performed with a single actuator. In the prototyped designs, however, four actuators are actuated simultaneously. Besides the actuators sharing the available pressure for inflation, the pressure loss over the connecting tubes and valves is not constant and some actuators have access to more pressure than others due to small inaccuracies of the valves and tube connectors. After identifying this, the assumptions are refined and designs are generated with simulated actuators with decreased speed and force. Unfortunately, this results in slow, less novel, crawling designs. The novelty of the pneumatic prototypes is low. Partly this is due to the limited capabilities of the actuators, partly this is because the designs selected to be prototyped are selected based on feasibility, not on novelty. To make the design of pneumatic soft robots more successful, the prototyping of the actuators has to be improved, e.g. through silicone 3D-printing.

The results of generating the multi-gait soft robot from literature suggest that designing pneumatic robots with strong actuators is possible with the proposed design method and results is performance prediction accurate enough for concept selection.

The performance of the tendon actuators is far better. Although there is still a large discrepancy between simulated and real speed, the prototypes have speeds comparable with soft robots from experimental robotic research and they outperform the majority with respect to novelty. The discrepancy in performance is expected to decrease when the issue of the power transportation is resolved.

As can be seen from the performance ratios of the experiments, no constant ratio of real performance to simulated performance is found. However, the ratios of designs within the same experiment are very similar, as is the case for the two simulated gaits of Shepherd's design in Section 5.4. This suggests that the inaccuracy for each type of prototype is systemic, indicating that modeling and prototyping more accurately could improve the ratios. It also means that the simulated performance can be used for a fair prediction of the real performance when the ratio for the intended type of robots is known.

To obtain successful prototypes, modeling and prototyping are to be viewed as an iterative process. Insights from prototyping, e.g. a re-evaluation of actuator strength, are used to improve the simulation model, generation method and prototyping method, thus improving the design process and the resulting designs.

The work in this chapter also shows that fabricating soft robots is still requires trial-and-error and is an art. The proposed CDS method can generate viable conceptual designs but is it up to the designer to detail and prototype the designs, requiring expertise. By providing a viable design concept, the CDS approach can possibly speed-up this process.

Finally, a direction of future research is proposed. The main limitation of the tendon-driven prototypes presented here is the transmission of actuation from the motors to the actuators. Due to friction and play in the thread guidance, the thread displacement at the actuator is less than the displacement at the motor and the guiding tubes exert forces on the design. Scaling up the robot approximately four times would allow space for the motors on the robot itself. This would result in a fixed position of the motors with respect to the actuators. The weight and size could be added as a rigid part in the generation process. The robot would still be tethered, however, it would only require a set of lightweight electrical wires running from the stationary Arduino and DC power converter to the servo motors.

6.5 SUMMARY

In this chapter, the detail design and prototyping of several conceptual designs resulting from the generative design method are presented. Two actuation methods are investigated, pneumatic actuation and tendon-driven actuation, without major modifications to the generation method and the simulation model. The tendon-driven prototypes are shown to be novel and fast compared to a collection of state-of-the-art soft robots. Although there is a clear discrepancy between the simulated and actual performance of the robots, the ratio between the simulated and actual performance is similar for prototypyes of the same type.

7

DISCUSSION

This chapter discusses the results presented in the previous chapters with respect to the objective and research questions stated in Chapter 1 and clarifies the contributions.

Since designing soft robots by hand and using a trial-and-error process is time-consuming and often limited to biological analogy, a generative design method with integrated simulation is presented. Additionally, there is a need for a method to compare the merit of conceptual designs. Therefore, the objective of this research is *to develop a computational design synthesis method for the conceptual design of the combined morphology and actuation of soft robots for locomotion*.

To achieve this goal, a CDS method is developed and an overview of the method proposed is shown in Figure 7.1. In the following sections, the answers to the research questions are discussed. The generation method (RQ 1.1) is evaluated first, followed by the simulation method (RQ 1.2) and the optimization method (RQ 1.3). The optimization of the actuation (RQ 2.1 and RQ 2.2) and the prototyping of generated designs (RQ 3.1 and RQ 3.2) are addressed subsequently. The last section discusses as to what extent the thesis objective as a whole is achieved.

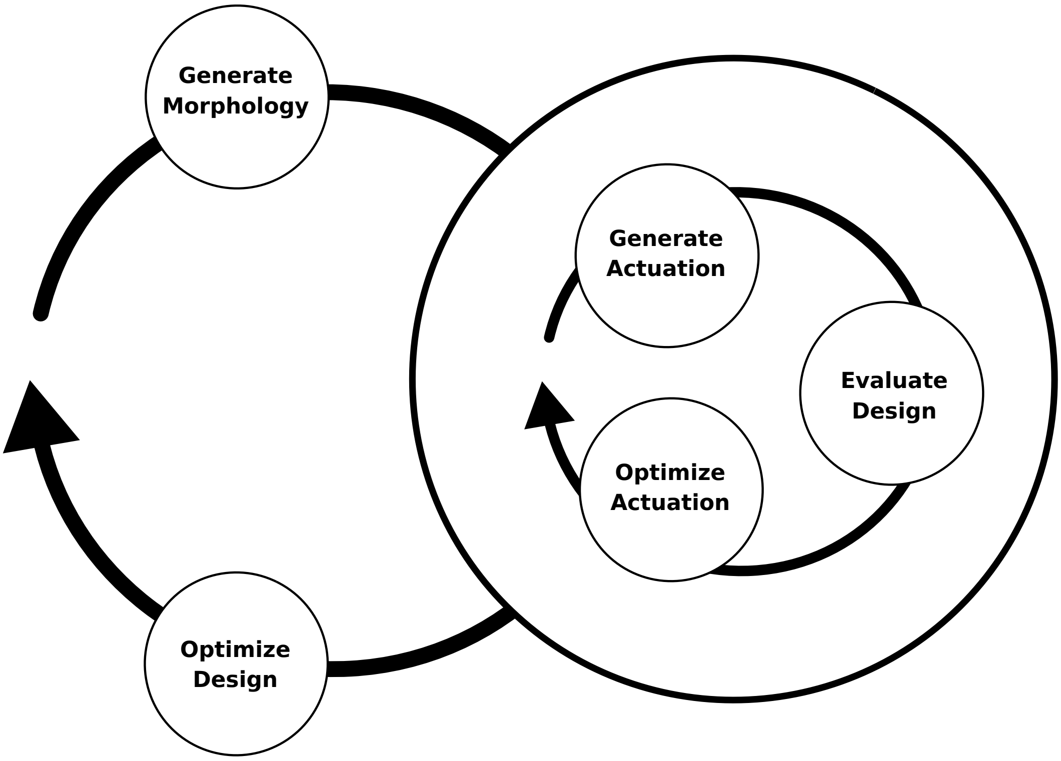


Figure 7.1: Overview of the CDS method presented in this thesis.

7.1 RQ 1.1: MORPHOLOGY GENERATION

How can the morphology generation process be controlled and guided, while still maintaining enough design freedom?

Generating soft robots with a spatial grammar is proposed in Chapter 4. The spatial grammar allows for the incorporation of design requirements and constraints directly, satisfying them in the generation stage rather than the evaluation stage. In Chapter 5, a refined spatial grammar is presented. This spatial grammar is used in Chapter 6 to generate both pneumatic designs and tendon-driven designs, with only minor modifications to the modeled constraints. Additionally, it is shown that observations during detail design and prototyping can easily be integrated in the grammar rules, e.g. pneumatic designs are restricted to have actuators connected to non-actuators only and tendon-driven designs can have positive and negative curvature. However, it is difficult to evaluate the quality of a spatial grammar for two reasons.

First, whether a balance is struck between generation guidance and design freedom is largely subjective. To judge the level of guidance, a designer determines if he or she can detail and fabricate the generated designs with the planned fabrication method. To judge if the method allows for enough design freedom, the variety and the novelty of the designs is decisive. Although a large variety of designs result from the proposed generation method, there are designs that can not be generated, e.g. a millipede. To analyze the novelty, a novelty metric is applied in Chapter 6. However, a certain degree of subjectiveness is unavoidable. Since the measures of guidance and freedom are matter of perspective, the presented method gives a designer the possibility to find the balance for him or herself through the design of the spatial grammar rules. However, this also means that for successful design generation, the designer requires some experience in constructing a spatial grammar.

Second, a spatial grammar also introduces a design bias. There are three ways in which this becomes apparent. First, some designs can be found more easily, i.e. with less rule applications, then others. With each rule application, the solution space is increased and the possible designs increase vastly, which means that designs that require more rule applications become possible only when the solution space is very large and therefore have a smaller probability of being generated. Second, the rules can render it infeasible to generate certain morphologies. This is exploited to exclude infeasible designs, however, the rules also exclude unknown designs with unknown performance.

Third, the size of the shapes added and removed by the rules of the proposed spatial grammars is independent of the size of the design. This means that applying a rule to a large design results in a proportionately small design change, while applying the same rule to a small design result in a proportionately large design change. Consequently, when a simulated annealing process accepts a large design that is a local maximum, possibly more rule application that do not improve the design need to be accepted to escape the local maximum than would be needed if the design were small. A research direction to explore further is to adapt the spatial grammars such that they can modify designs depending on their size, possibly through design size dependent rule parameters or rules that can only be applied to designs of a certain size.

The designs resulting from the CDS method presented in Chapter 4 show a large variety of morphologies and gaits that is not seen in physical soft robot design. Although this suggests that the solution space of soft robots for locomotion holds more than what is prototyped, it is possible that the novelty of the design is enabled only by the absence of fabrication constraints. The same can be said about the generated soft robots presented by [25] and [45]. This work, however, also presents the CDS of soft robots that can be prototyped with common actuation and fabrication methods in Chapter 5. The resulting designs show that the solution space of soft robots for locomotion with common actuation methods spans more than the soft robots that currently exist.

Contributions

The first spatial grammar for the generation of soft robots is formulated. As opposed to the black-box generation methods used in related work [25] to [45], design generation with a spatial grammar allows for a measure of design guidance, giving the designer control over the type of morphologies to be generated. By integrating fabrication constraints directly, the fabrication of generated designs with a planned actuation method is enabled.

The design of soft robots that can be prototyped with common actuation and fabrication methods demonstrates, for the first time, that the solution space of soft robots spans more than the soft robots that currently exist.

7.2 RQ 1.2: SIMULATION OF SOFT ROBOTS

Which simulation model best supports the evaluation of soft robots concepts for locomotion within CDS?

To simulate soft robot concepts, two simulation models are developed in Chapter 3. Both models are based on rigid-body dynamics and use an explicit simulation method. To minimize computational costs, they use a simplified design representation. The first model represented soft robots as collections of balls connected with springs. Actuation is achieved by manipulating the rest lengths of the springs. The model is based on the assumption that only bending actuators are simulated. It is stable for all generated designs and is used for the design of morphologies in Chapter 4. However, the ball-spring model is a weak approximation of reality. Therefore, the model is simplified further by modeling the motion of bending actuators, rather than the internal forces within the actuators.

Thus, the second model, a rigid-hinge model, prescribes the motion of simulated bending actuators to approximate physical bending of actuators directly. As is assessed in Chapter 6, this model suffices for conceptual design and, since it uses significantly less constraints for similar modeled actuators than the balls-spring model, it is computationally more efficient. However, it is not accurate enough for detail design and for this model to function, the designer is responsible for the modeling of the actuator motion. Thus, this model shifts some of the challenges of soft robot simulation back to the designer. A more accurate simulation model, and specifically a better friction model, might also allow for the generation of a snake-like robot, which is currently not possible.

Due to time limitations, the search for a simulation method conducted in Chapter 3 is not exhaustive. In addition, the research field of soft material simulation has advanced since then. A recent development in the simulation of soft robots is a physics simulator called Chain-Queen [66], a method based on the Moving Least Squares Material Point Method (MLS-MPM). The simulation is differentiable, which opens up the possibility to use a gradient-based optimization method to optimize the actuation. Further, collision is solved intrinsically through the MPM method. Since collision is one of the main challenges for the methods discussed in Chapter 3, this method deserves investigation.

The increased accuracy new simulation methods might offer can be used to transfer the design detailing, now carried out by the designer, to the CDS. More accurate simulation models are potentially computationally more expensive since they model the soft robot in higher detail. A staged design process could be used to take advantage of the different simulation methods. In a staged design process, the conceptual models presented in Chapter 3 can be used for the CDS of conceptual designs and a parametric optimization, using a more detailed model, can be used to optimize the detailed design.

Contributions

The two rigid-body simulation models presented here allow for the stable simulation of soft robots with legs, which is not shown by the simulation models presented by [25] to [45].

7.3 RQ 1.3: OPTIMIZATION

What are the characteristic of the solution space and which optimization method is suitable to carry out the optimization?

The processes described in Chapters 4 and 5 both use simulated annealing for the optimization of the designs. Here, three insights into the search space and the use of simulated annealing within this design process are presented.

First, the grammar rules introduce large changes in objective function values and thus create an optimization problem with a discrete solution space. Independent of the generation method, the solution space of locomotion robots presumably consists of a vast variety of morphologies with varying gaits. Thus, it is likely that the solution space is multi-modal. Three designs shown in Figure 7.2 and resulting from the CDS method presented in Chapter 5 confirm this. They have the same objective function value but have different morphologies and gaits.

Second, running multiple optimization processes gives an indication of what is likely the upper bound of performance for the specified design task and spatial grammar, however, there is no way to be certain the maximum is found. For the task of concept generation, however, it is helpful that different concepts are provided, as long as the performance of the concepts approximates the global maximum.

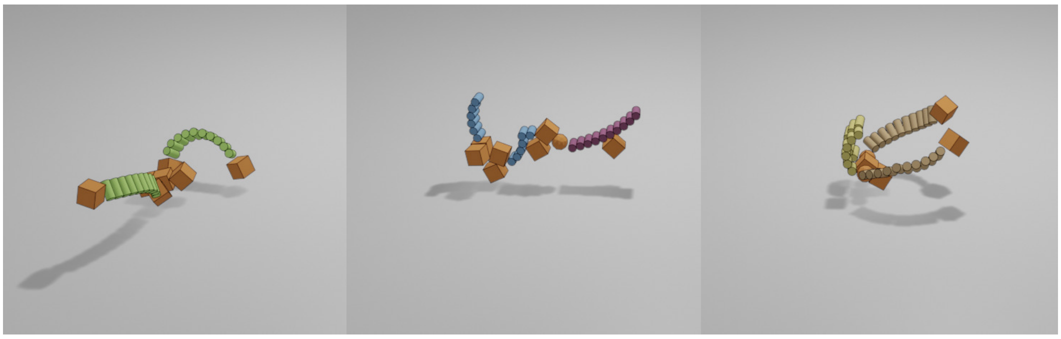


FIGURE 7.2: Three designs with the same objective function value yet very diverse morphologies and gaits. All three designs have an objective function value of 76.

Third, as described in Section 7.1, the design size independent rules make is more likely for the simulated annealing process to stall on a large design than on a small design. This complicates tuning the annealing temperature and renders more sophisticated annealing schedules impossible. This is another incentive for the research into scalable spatial grammar rules for the generation of soft robots. A second direction of research is a design size dependent annealing temperature.

Contributions

The insights provided by the analysis of the spatial grammar rules in conjunction with simulated annealing as an optimizer help outline ways to improve the effectiveness and efficiency of this combination, which is used for many different design tasks [61] [62].

7.4 RQ 2.1 AND 2.2: CO-DESIGN OF MORPHOLOGY AND ACTUATION

How can an actuation pattern be determined for an arbitrary design within a limited time frame? How can the search for a actuation pattern for every design candidate be integrated in the CDS method?

To enable the co-design of morphology and actuation, Chapter 5 investigates two RL methods and a PAC optimization method. The three methods are used to optimize controllers for designs previously generated with a design method for morphologies only. These experiments show that neither of the RL methods can be used to optimize the control of an arbitrary design without tuning the learning parameters for that given design. A computational design method of simple sail boats that integrates RL to learn the angle of the sails is presented by [50]. However, it is unclear why RL is chosen, how the parameter tuning is resolved and if the learning improves the result compared to using a constant or random angle.

Here, PAC optimization is selected for the inclusion into the CDS process. The resulting co-design of morphology and actuation is shown to outperform the CDS of morphologies with non-optimized actuation, which have pre-defined actuation patterns.

Despite PAC being a type of feedforward control and relying completely on the simulation model accuracy for its performance on a physical robot, it performs well for the co-design of concepts. For the control of physical robots in an environment with uncertainties, a robust controller with sensor feedback is always preferred. Here, however, actuation optimization is introduced to increase the solution space and remove the design bias introduced by the selected fixed actuation patterns. After finding a suitable morphology for the task at hand through co-design with PAC optimization, a RL method could be used to optimize a controller for the physical robot.

Lastly, if a differentiable simulator is successfully integrated in the design process, a gradient-based method like optimal control can be used. It is expected to be far more efficient and outperform all three tried methods.

Contributions

This work discusses the criteria for the selection of an integrated actuation optimization method. A rationale for the selection of PAC optimization is given and it is shown that co-design outperforms the design of morphologies without actuation optimization with respect to both effectiveness and efficiency. The larger variety of morphologies and gaits resulting from co-design demonstrates the expansion of the solution space.

7.5 RQ 3.1 AND 3.2: PROTOTYPING

How can the design concepts be detailed and prototyped?
How can insights from the process of prototyping be integrated directly in the CDS method?

Detailing and prototyping of generated designs is shown in Chapter 6. It is demonstrated how two different types of soft robots, pneumatically actuated and tendon actuated robots, can be designed with a single CDS method through minor adjustments to the spatial grammar constraints and the simulation model. Although the prototypes do not reach the speed predicted by the simulation, they are fast enough to compete with the state-of-the-art soft robots.

The prototyping of seven generated designs reveals two things.

First, design and fabrication are to be viewed as an iterative process. Within every design process, assumptions are formulated, sometimes consciously and sometimes unconsciously. Only during fabrication and experimentation, possible invalid assumptions become apparent. By adjusting and improving the design method based on the performance of the prototypes, designs of a consecutive iteration are able to perform better.

Second, detail design and prototyping soft robots requires experience. Even when a fabrication method from literature is used, the craftsmanship needed to bring the fabrication to a success is developed over time. Ideally, additive manufacturing will mitigate this problem in the future, however, currently successful 3D-printing and assembly still require experience and skill.

Contributions

Here, the first automatically generated physical soft robots are presented that can compete with state-of-the-art robots with respect to speed and novelty. Four pneumatic robots and 3 tendon-driven robots are presented. Additionally, it is shown that observations in the protoyping phase are incorporated in the design method to improve the resulting designs.

7.6 THESIS OBJECTIVE

This last section discusses the main objective of the thesis: *to develop a computational design synthesis method for the conceptual design of the combined morphology and control of soft robots for locomotion.*

A large variety of morphologies and actuation patterns are generated using the proposed CDS method, providing a designer with choice; see for example the equally performing designs in Figure 7.2. Highly novel design concepts, which are feasible to fabricate and perform reasonably well, are generated for the type of actuation and fabrication selected by the designer.

To show where this research is located with respect to related work, the two main differences for the two most relevant references are presented in Table 7.1.

As can be seen, the work presented here fills a gap in the field since related work lacks the following two characteristics. First, the possibility to guide design towards designs that more closely mimic the state-of-the-art soft robots facilitates the design detailing and prototyping done by a designer. A CPPN method for the generation of designs [27] is a black-box method that does not allow the direct incorporation of design and fabrication constraints, as is the case for the L-system generation method presented in [45]. The graph grammar presented by [47] generates rigid robots and the work focuses on generating virtual creatures. These methods do not take into account constraints set by prototyping.

Second, of the three referenced works, only [27] present an attempt at prototyping a generated design. However, the actuation of the voxel-based design, consisting of closed-cell and open-cell voxels, is achieved by lowering the pressure in a pressure chamber.

In short, where other methods show interesting virtual creatures, this work designs soft robots in an engineering context that can be successfully prototyped with common actuation and fabrication methods.

A limitation of this work is that designs are restricted to bending as an actuation method. Although this is the predominant actuation method for physical soft robots, soft robots using extending actuators is considered a direction that deserves exploring.

Also, in this work, there exists a strong relation between the generation method and the simulation model. The spatial grammar rules modify modeling elements of the simulation model. As a consequence, it is not straightforward to exchange the simulation model with

TABLE 7.1: This design method proposed in the work in comparison to related work on the automated design of soft robots.

	This work	Cheney et al. [27]	Rieffel et al. [45]	Sims [47]
Design guidance / engineering context	Yes, through a spatial grammar	No, CPPN generation	No, L-system like generation	No, graph grammar for rigid robots
Prototyping of designs	Yes, pneumatic and tendon actuation	Yes, actuation through environmental pressure	No	No

a different simulation model or method. This dependency between generation method and simulation model is not only a limitation of this work, it is also found in referenced work. For example, the voxel-based simulator used by Hiller et al. [52] is combined with a CPPN that generates voxel-based designs. To model designs with extremities with arbitrary orientation, a larger voxelized design space is required. This is likely to be computationally infeasible.

Future research in the field of automated design of soft robots should move aim for a process where the different components can be exchanged independently.

Contributions

The CDS method presented in this thesis is versatile and synthesizes a wide range of soft robot designs for, at least, two differently actuated types of robots.

The designs demonstrate that the solution space of soft robots for locomotion with common actuation and fabrication methods spans more than the soft robot designs shown in literature and for many of them a biological analogy is hard to find; see Figure 7.3 for four examples.

By presenting a designer with different design with equal performance, it is shown there is no single best solution and the designer has choice.

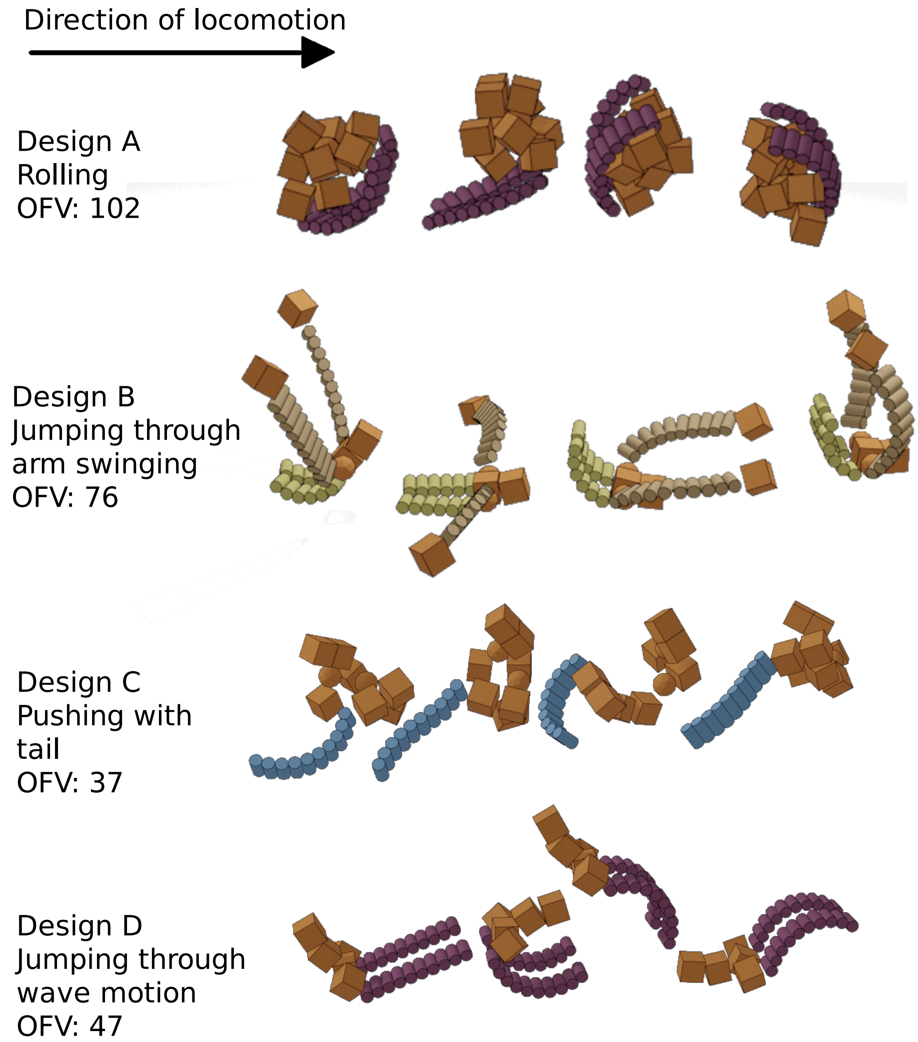


FIGURE 7.3: Four designs resulting from the CDS method presented in this thesis that show locomotion for which a biological analogy is hard to find. For each design, the gait is described briefly and the objective function value (OFV) is given.

8

CONCLUSION AND FUTURE WORK

The field of soft robotics is steadily growing and an increasing number of industrial applications is found, e.g. in soft grippers [15]. However, there is a need for a method to explore the wider solution space, optimize designs and enable an informed choice among soft robot concepts.

To overcome design bias and allow for the comparison of different conceptual designs, this thesis proposes a CDS method for the design of soft robots for locomotion.

First, two spatial grammars are investigated (RQ 1.1) for the generation of soft robot morphologies in Chapter 4. The generated designs are simulated (RQ 1.2) with one of the two rigid-body models presented in Chapter 3 and optimized (RQ 1.3) with simulated annealing. Second, three actuation optimization methods are explored (RQ 2.1) and the design of morphologies is extended by optimizing the actuation of every morphology (RQ 2.2) with PAC optimization in Chapter 5. Third, in Chapter 6, generated designs are prototyped (RQ 3.1) as pneumatic and tendon-driven robots and it is discussed how the design process is improved through insights resulting from the prototyping process (RQ 3.2).

The contributions of this work can be summarized as follows:

- The first CDS method for the design of soft robot morphologies for locomotion that explicitly models design requirements and constraints, including those from fabrication, giving designers a measure of guidance.
- Two rigid-body approximation models for the simulation of soft robots with bending actuators are implemented and shown to allow for the stable simulation of all generated soft robots, even the designs with legs.
- PAC as a method for the optimization of the actuation patterns within a CDS method is presented and is shown to balance computational cost and performance, while not needing design specific tuning.
- The co-design of morphology and actuation results in designs with higher performance and larger variety than the designs resulting from the design of morphology only.
- The prototyped designs compete with state-of-the-art soft robots and are highly novel according to the used novelty metric.
- The generative design method is shown to be versatile, since it can generate concepts for soft robots with different actuation methods with only a few modifications to the modeled design constraints.

- The prototypes of the generated designs demonstrate that the solution space of soft robots for locomotion includes designs not presented before, without introducing new actuation and fabrication methods. Additionally, it includes designs without a clear biological analogy and it is demonstrated that multiple equally performing solution exist.

There are several limitations to this work. First, the construction of a successful spatial grammar requires a measure of experience on the side of the designer and it is hard to assess if the designs excluded by a grammar are all indeed infeasible and undesirable, i.e. which unknown unknowns are excluded? Second, the PAC optimization assumes cyclic actuation and thus this method can not be generalized to non-cyclic design tasks. Third, the simulation models presented here are not accurate enough for detailed design. As a consequence, detailing the designs is left to the designer. A more accurate simulation model would allow designers without experience in soft robot design to prototype the generated designs. The use of a differentiable simulation model would allow for the use of a gradient-based control optimization, e.g. optimal control. This would possibly extend the co-design method to non-cyclic design tasks. Fourth, the combination of design size independent spatial grammar rules and simulated annealing creates challenges for the tuning of the annealing temperature and the annealing schedule. A possible approach to mitigate this is constraining the application of certain spatial grammar rules to certain design sizes or introducing design size dependent rule parameters. Last, the generation method and simulation model in this design method are strongly dependent, which makes exchanging the methods used for the components of the CDS independently difficult.

Last, several possible extensions of the work are proposed. An interesting next step is the CDS of untethered soft robots. Integrating the power source of the robot in the body not only solves many of the problems encountered with protopying in Chapter 6, it would also bring the robots closer to functional robots. When pneumatic or tendon actuation as implemented here are to be used, the robots would have to be scaled up to accommodate the power source. The presented CDS method can be adapted to generate larger robots with a rigid block representing the power source. An open question is how the scaled up robots are to be fabricated, since the 3D-printer used for the prototypes here would be too small. Recent advances in the printing of silicone could resolve this. Second, extending the task by including direction would increase the practicality of the design in real applications. By evaluating a morphology and two sets of PACs for two separate objective value functions, one for moving forward and one for going left or right, a morphology can be optimized that can change direction. Thus, the evaluation phase will consist of two actuation optimization and two evaluation components. It is interesting to investigate if this will result in a separate set of actuators dedicated to changing direction. A similar approach can be used to extend the CDS method to generate multi-task robots. The two objective value functions would then be used to evaluate different tasks, for example locomotion and gripping.

In conclusion, the designs resulting from the presented co-design CDS method span a wide range of morphologies and gaits, many without biological analogy. Multiple solutions are generated with the same objective function value showing the multi-modality of the solution space. This work shows that the CDS of morphology and actuation enables a wider exploration of the entire solution space compared to bio-inspired design alone.

A

SOFT-BODY DYNAMICS

In this appendix, the experiments with force-based soft-body dynamics and position-based soft-body dynamics are presented. These methods are investigated as part of the search for a suitable simulation method as described in Chapter 3.

A.1 FORCE-BASED SOFT-BODY DYNAMICS

To examine the practicality of the spring-mass model implementation of BPL for the evaluation of the soft robot designs for locomotion, two types of soft robots are modeled and evaluated.

The first designs examined are pneumatically actuated and consist of actuated soft material and unactuated rigid material, see Figure A.1a for an example of such a design. An assembly of inflatable hollow cubes is created to represent the soft material of a design. The cubes are modeled as surfaces consisting of point-masses connected with elastic distance constraints. Rigid material is modeled as using the standard impulse-based dynamics modeling. The soft material is connected to the rigid material with stiff distance constraints. To represent the internal pressure following pneumatic actuation, the pneumatic actuation is added by subjecting all point-masses belonging to an actuated surface to a force perpendicular to the face they are part of. Solving this model with the BPL shows that modeling material as a hull fails to represent a materials resistance to bending. As a result, the inflated hull can crumple at collision and does not unfold properly anymore. Therefore, this model is abandoned.

The second type of design consists of similar cubes of soft material, however, distance constraints inside the cube are added, i.e. these cubes are no longer hollow. Rigid material is modeled the same as soft material, but with stiffer springs. Instead of modeling the physics behind the actuation, now, the intended behavior is modeled. As will be discussed further in Section 4.2.1 when the spatial grammar for generation is presented, the generated designs use bending actuators. Therefore, the model can be manipulated to bend the actuators directly. It is achieved by modifying the rest-length of the distance constraints between the point-masses. When the rest-lengths are increased on one side of an actuator and decreased on the opposite side, the actuator will bend. In Figure A.1b a design with bending actuators modeled this way is shown.

As can be seen from Figure A.1, both models use voxels, or cubes, to build a design. The rationale for this choice is that this simulation method creates a relation between mesh resolution and material stiffness. BPL uses an iterative constraint solver (Gauss-Seidel) and this causes local constraint satisfaction. What this means is that, within one constraint solver iteration, only points in the direct neighborhood of a point that moved are effected, see

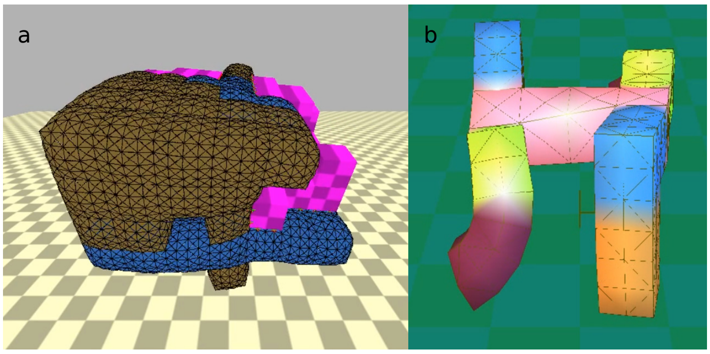


Figure A.1: Using soft body dynamics from the Bullet Physics Library to simulate (a) inflating designs through surface forces and (b) bending designs through distance constraint manipulation. The colors in both pictures signify material stiffness and actuation.

Figure A.2 for an example. Consequently, the resolution of a mesh defines how many solver iterations are needed for the error correction to propagate through the whole mesh. This, in turn, defines the actual material stiffness. So by generating voxel-based designs, the mesh resolution is kept constant everywhere and therefore the material stiffness is kept constant everywhere.

As long as the time step and the velocities are small, the overshoot resulting from the explicit integration and sequential constraint solving is small. However, complications occur with the stability of contact forces. When a point penetrates a surface, a temporarily constraint modeled as a spring is added to the system. The added constraint connects the penetrating point to the face it penetrates. The penetration depth, which is dependent on the time step size and the velocity of the penetrating point, determines the spring-stiffness. The point is pushed 'back out' but has an unnatural acceleration as a result. This can result in designs that 'fly off' after a collision or in unactuated designs that magically hover or jitter over the floor.

A second issue is the tuning of the spring stiffness. A low stiffness creates very soft material that is unable to support its own weight. A high stiffness, however, can create instability by resonating. Finding a balance between the two is very challenging and might even be infeasible. Both problems can be mitigated by decrease of the time step or increase of the number of constraint solver iterations, both, however, are computationally very expensive.

In summary, the BPL implementation of force-based spring-mass models can be tuned for the simulation of single designs but is not tractable for the evaluation of designs within this generative approach.

A.2 POSITION-BASED SOFT-BODY DYNAMICS

A brief overview of the PBD algorithm [76] is given here.

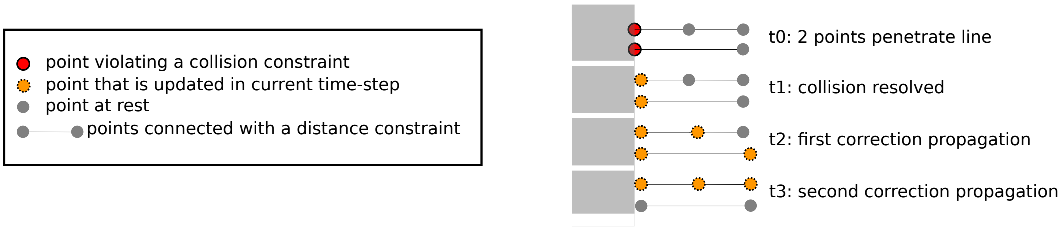

point violating a collision constraint
point that is updated in current time-step
point at rest
points connected with a distance constraint
t0: 2 points penetrate line
t1: collision resolved
t2: first correction propagation
t3: second correction propagation

FIGURE A.2: Two line segments with different resolution illustrate the difference in error correction propagation. Each segment consists of 2 or 3 vertices, connected by springs (shown as lines). Here t is the time-step, or constraint solver iteration step. The two red vertices that penetrate the wall are in violation of the collision constraints. Their collision constraint violation is solved in the first constraint solver iteration. Their movement is only noticed by neighboring constraints during the consecutive iterations. The low resolution segment updates the complete mesh in less iterations. Therefore, the low resolution segment will behave stiffer for the same number of constraint solver iterations.

This method solves constraint explicitly, resulting in the same relation between mesh resolution and material stiffness as discussed in Section 3.2.1. To achieve a relatively stiff body, one can opt for a small time step, many constraint solver iterations or a coarse mesh. None of these three options are desirable. Macklin et al. [94] address this problem of iteration count and time step dependent constraint stiffness. However, what is also implied by this relation is that mesh areas with different resolutions will have different material stiffness.

A set of vertices p that forms an object are subjected to a set of constraints governing the distances between these vertices. A distance constraint between vertices p_1 and p_2 is an equality constraint of the following form

$$C(p_1, p_2) = |p_1 - p_2| - d \tag{A.1}$$

where d is the distance between the vertices at initialization, i.e. the rest length. Every simulation time step consists of two parts. First, the velocities are updated based on the external forces and the positions are updated based on their velocities. Second, the positions are updated to meet the constraints using constraint solver iterations. The position update Δp_1 for vertex p_1 for the distance constraint given in A.1 is given by

$$\Delta p_1 = k(|p_1 - p_2| - d)\frac{p_1 - p_2}{|p_1 - p_2|} \tag{A.2}$$

for vertices with an equal mass and constraint stiffness k. For the update of the position of vertex p_2 holds: $\Delta p_2 = -\Delta p_1$. As mentioned in the previous section, the number of constraint solver iterations per time step has a large influence on the effective stiffness of the distance constraints and on the displacement propagation through the object. The effective stiffness is also influenced by the mesh resolution, see Figure A.2 in the previous section. Since a higher resolution means a larger number of distance constraints per unit length of mesh, more updates to propagate the displacements over the whole mesh are necessary.

A.2.0.1 *Implementation of Friction, Contact Resolution and Actuation*

First, implementing friction as a dampening factor on the velocity of vertices that have come into contact with a surface is tried. This proves to be insufficient since the vertex movement of vertices that are close to a surface is predominantly caused by collision constraint resolution, which is independent of the velocity of a vertex.

Therefore, friction is added to the simulation model in the form of constraints, similar to the distance constraint given by equation A.1. In equation A.3, the update for a kinetic friction constraint is given.

$$\Delta p^t = \mu_k (p^{t-1} - p^t) \tag{A.3}$$

Here Δp is the vertex position update, p the vertex position at time t and μ_k the kinetic friction coefficient. The friction constraint updates are independent of the normal force of the vertex on the surface because this force is unknown.

The detection and resolution of collision is based on discrete vertex-face and face-face collision detection. For the vertex-face collision detection, the relative positions of vertices with respect to faces is used to detect penetration. A vector is constructed from the current relative location to the old relative location and the ray-triangle intersection algorithm as described by Möller and Trumbore [95] is used. This is illustrated in Figure A.3 where the relative position r_{t0} of point v_{t0} with respect to the face $p1, p2, p3$ is determined at time-step t_0. At time-step t_1, a ray is constructed from the former relative position r_{t0} to the current point position v_{t1} and checked for collision with the face. In case of intersection, a constraint that projects the vertex on the surface of the face is generated, either on the positive or negative side. When a multitude of intersections is found, the vertex is projected on the surface of the first face it penetrated. To reduce the number of calculations, a hash table is used.

The stability of this collision solving approach relies on the assumption that the detection is absolute. If the penetration of a face by a vertex is missed, e.g. when the distances are close to machine precision, and in the consecutive time step the vertex returns to the correct side of the face, the algorithm might falsely project the vertex back to the wrong side. A second complication with this approach is the intersection of four faces, without any vertices having penetrated a face, see Figure A.4. Therefore, the triangle-triangle (also called face-face) detection algorithm described by Guigue and Devillers [96] is used. Although every point-triangle collision also results in a triangle-triangle collision (note that the opposite is not true), resolving the point-triangle collision is more straight-forward, and thus a combination of these methods is used. In the case of a pure triangle-triangle collision, the vertex positions are updated in the direction of the old positions which resolves the collision, but is not a realistic response.

Also here, first an approach is tried that approximates inflation by assigning an acceleration to the vertices in the normal direction of their faces. However, by defining the actuation as a prescribed acceleration, it is not part of the constraint solving iterations. Thus, by increasing the number of constraint solver iterations, the point positions are updated to meet

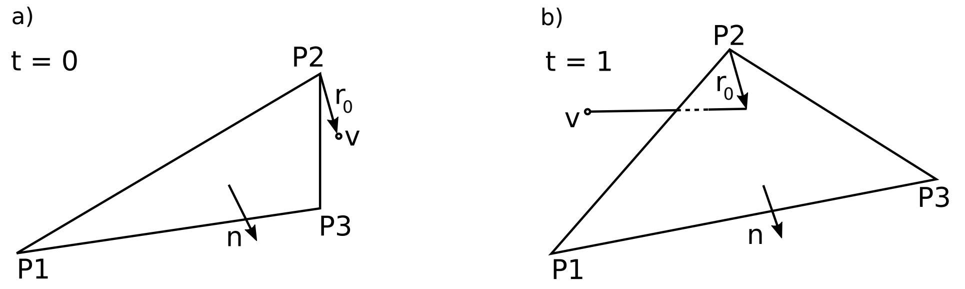


FIGURE A.3: Constructing the ray to be checked for Ray-Triangle Intersection. a) Determining the relative position of point v with respect to the face p_1, p_2, p_3 at time-step t_0. The relative position is denoted by r_0. b) Defining the ray r_0 - v to check against face p_1, p_2, p_3 at t_1.

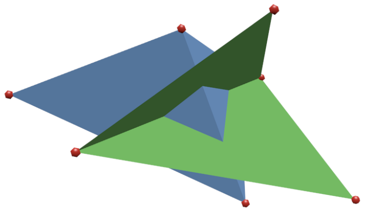


FIGURE A.4: An example of triangle-triangle collision without point-triangle collision.

the constraints more often and the inflation effect is diminished. Second, the accelerations are introduced before the constraint resolution step. As a consequence, the direction of the acceleration is set before the points are repositioned to meet the constraints.

Hence, activation of the mesh is achieved by changing the rest-lengths and the stiffness of the distance constraints between the vertices. The rest-length of a spring is the length at which the spring does not exert any force. An additional benefit of implementing actuation as change of the springs rest-lengths is that there is no additional computation needed to accumulate the face normals a point belongs to in order to find the acceleration direction.

A.2.0.2 *Results*

The position-based dynamics library from [80], extended with collision resolution, friction and actuation, is able to simulate a large variety of designs. In Figure A.5, three designs that result from a generative design process are shown as illustration. However, the contact resolution method causes jumps and jerking motions when the time step is too large, especially when multiple face are penetrated by a single vertex within one time step. When a soft robot buckles or crumples, points cross a face from both side, resulting in an additional challenge of defining the side of the face that a point came from. Using only local mesh-data, i.e. data about the surrounding vertices and faces, forces the use of temporal data to define where a point came from. In case a point is moving very slowly, computer precision prevents the correct calculation of the velocity. After a failed detection, correct detection in subsequent time-steps will maintain the error for the remainder of the simulation.

Modeling the designs as meshes imposes several restrictions. The condition that faces are triangular, imposed by the ray-triangle and triangle-triangle collision detection, increases the mesh resolution, which in turn increases simulation times. A voxel-based system is used for the genartion of the designs so the meshing can ensure uniform faces over the entire design. Of course, this shapes the morphologies. An alternative for modeling with meshes is modeling the designs with clusters of particles repelling each other, see [79].

In summary, PDB solves several of the collision resolution problems encountered in force-based dynamics. However, it has become clear that adequately implementing collision resolution is outside of the scope of this research.

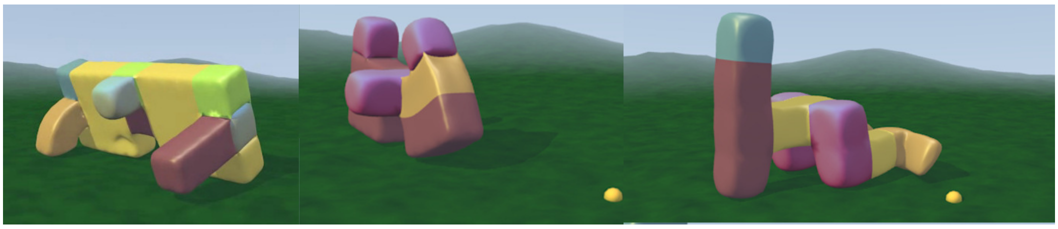


FIGURE A.5: Walking robot designs simulated using position-based dynamics.

B

DEEP Q-NETWORKS REINFORCEMENT LEARNING

In this appendix, the results of Double Deep Q-Networks (DQN) as control optimization method are discussed. The method is implemented and investigated as part of the search for a suitable actuation optimization method as described in Chapter 5. DQN is an off-policy, discrete, DRL method that is best known for outperforming humans on ATARI games [97], [98], [99] and is also used to control a robot arm [100].

DQN is used to optimize a controller for Design A; see Figure 5.4 in Chapter 5. For more stability, a double DQN method [99] is selected and implemented. The method can be summarized as follows.

To solve the action selection problem, Q-values are learned. A Q-value is the expected sum of future rewards when taking action a now and taking optimal actions in the future. In DQN, a neural network is trained to estimate the Q-values. Every simulation time-step, the current state s_t of the robot is provided to the input of the Q-net, a multi-layer neural network with weights θ_t. The output nodes of the Q-net return estimates of the Q-values $Q_\pi(s_t, a_t)$ for each possible action a_t at time t, given the current policy π. With a probability of ϵ, a random action is selected, the action with the highest estimated Q-value is selected otherwise. To improve the estimated Q-values, experiences from simulation are collected. An experience is a set containing a state s_t, an action a_t and a reward r_{t+1}. In double DQN, as opposed to DQN, a second Q-net, called the target network with weights θ'_t, is used in the process of training the Q-net weights. This target network only differs from the Q-net with respect to the values of the weights.

The error η of the Q-net is determined by:

$$\eta = \alpha(r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_t, a; \theta_t); \theta'_t) - Q(s_t, a_t; \theta_t)) \tag{B.1}$$

Where, α is the learning rate and γ is the discount rate. As Equation B.1 shows, the action is selected according to the estimated of the Q-net but the value of this state-action pair is determined according to the estimate of the target network. The update of the Q-net is given by:

$$\theta_{t+1} = \theta_t + \eta \nabla_{\theta_t} Q(s_t, a_t; \theta_t) \tag{B.2}$$

Either, the weights of the target network are replaced by the weights of the Q-network at regular intervals, or they are updated every time the Q-net is updated according to:

$$\theta'_t = (1 - \tau)\theta'_t + \tau\theta_t \tag{B.3}$$

where τ is the target update parameter. Training a neural network with a moving target is difficult. Both the experience buffer and the target network aim to make learning more stable by making the target less erratic.

To implement DQN for the locomotion task, a reward function is formulated. It is tuned specifically for the task, however the objective function value as given by Equation 4.3 is used to compare results to the performance of the design without optimized actuation. A sparse reward function that places a milestone every 10 units from the origin is constructed. Reaching a milestone results in a reward of five. Every milestone only produces a reward once, so repeatedly passing the same milestone is not rewarded.

As for SAC, the input state s_t of the Q-network consists of four components illustrated in Figure 5.8 and described in Section 5.2.2. The components are normalized and are the input of a Q-network with two hidden layers, with respectively 18 and 9 nodes. The output layer has one node for every combination of actions. To allow a selected action to have a measurable effect, a control step, or RL state transition, occurs only three times per second. Every control step, each actuator executes one of three actions; incrementing the target angle by 0.3 radians, decrementing it by 0.3 radians or keeping the angle constant. For two actuators, this makes the total amount of output nodes $3^2 = 9$. The ϵ value for ϵ-greedy action selection starts at 0.9 and gradually decreases to 0.05. An ADAM optimizer [101] updates the Q-network weights to better fit the agent's experience, i.e. minimize the error.

The mean best objective function value and the range of best objective function value plotted against the number of state transitions for 21 runs is shown in Figure B.1. Table B.1 gives an overview of the performance. This demonstrates the need for integrated actuation optimization in the CDS.

Experiments with more than two actuators do not produce good results. Increasing the number of nodes per network layer is investigated but does not improve the performance further. DQN is not effective for problems with more than two actuators; as is also found by [102] and [103]. The problem is explained by looking at the size of the action space. In a discrete action space, experience for each action is to be gathered for the agent to learn. The symmetric designs generated here have two, four, six or eight actuators. A design with four actuators, each with three actions, already has 81 distinct actions the robot can take. Given that DQN is not practicable for the control optimization of designs with four or more actuators, it is not investigated further.

TABLE B.1: The DQN results compared to the fixed actuation result for Design A.

Design	Fixed Actuation Result	Best Result / Fixed Actuation Result	Worst Result / Fixed Actuation Result	Runs Better than Fixed / All Runs
Design A	80.0	1.25	0.97	20/21

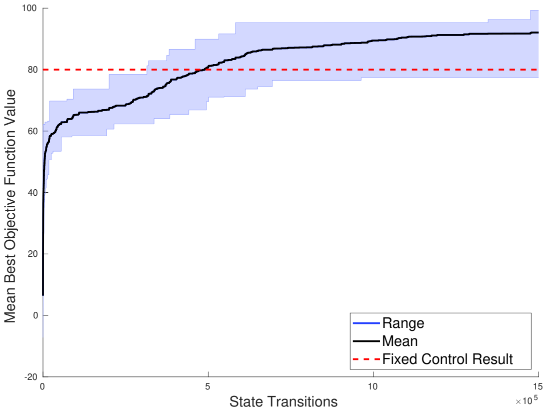


Figure B.1: Mean best objective function value versus state transitions of 21 runs of DQN for Design A. The objective function value of the design with fixed actuation is shown in red dashed line.

C

NOVELTY OF STATE-OF-THE-ART SOFT ROBOTS

In this appendix, the collection of soft robots used for the novelty score calculation in Chapter 6.3.1 is presented. Robots with multiple modes of locomotion are counted as multiple robots, e.g. the two robots by Tolley et al. have the same morphology but differ in gait.

TABLE C.1: The collection of soft robots used for the novelty score calculation in Chapter 6.3.1.

Robot	Number of actuators	Gait	Symmetry	Maximum number of connections of an actuator	Actuators per limb	Bi- or unidirectional bending	Novelty score
Tolley et al. [10]	4	Undulating	2 axis	1	1	Uni	17.3
	4	Walking	2 axis	1	1	Uni	22.0
Shepherd et al. [9]	5	Undulating	2 axis	2	1	Uni	26.0
	5	Walking	2 axis	2	1	Uni	30.6
Onal and Rus [22]	4	Snake	1 axis	2	Other	Bi	43.3
Lee [92]	6	Undulating	2 axis	2	1	Uni	27.3
Mao [21]	1	Undulating	1 axis	1	1	Uni	20.7
	3	Undulating	1 axis	1	1	Uni	20.7
	4	Undulating	2 axis	1	1	Uni	17.3
	5	Undulating	1 axis	1	1	Uni	19.3
Du Pasquier et al. [39]	4	Undulating	1 axis	2	1	Uni	15.3
Design A	4	Walking	1 axis	1	1	Uni	20.0
Design B	4	Undulating	1 axis	1	1	Uni	15.3
Design C	4	Sideways walking	1 axis	1	1	Uni	21.3
Design D	4	Undulating	1 axis	1	1	Uni	15.3
Design E	1	Other	1 axis	1	1	Bi	36.7
Design F	3	Worm crawl	No	2	1	Bi	43.1
Design G	4	Jumping	1 axis	2	2	Bi	40.6

BIBLIOGRAPHY

1. Haddadin, S., Albu-SchäCurrency Signffer, A. & Hirzinger, G. Requirements for safe robots: Measurements, analysis and new insights. *International Journal of Robotics Research* **28**, 1507 (2009).
2. Brown, E., Rodenberg, N., Amend, J., Mozeika, A., Steltz, E., Zakin, M. R., Lipson, H. & Jaeger, H. M. Universal robotic gripper based on the jamming of granular material. *Proceedings of the National Academy of Sciences* **107**, 18809 (2010).
3. Deimel, R. & Brock, O. A compliant hand based on a novel pneumatic actuator. *Proceedings - IEEE International Conference on Robotics and Automation*, 2047 (2013).
4. Petković, D., Pavlović, N. D., Shamshirband, S. & Anuar, N. B. Development of a new type of passively adaptive compliant gripper. *Industrial Robot* **40**, 610 (2013).
5. Goswami, A. & Peshkin, M. A. PASSIVE ROBOTICS : AN EXPLORATION OF MECHANICAL COMPUTATION Locus of attainable centers of, 1 (1990).
6. Wang, J.-Y. & Lan, C.-C. A constant-force compliant gripper for handling objects of various sizes. *Journal of Mechanical Design* **136**, 71008 (2014).
7. Whitesides, G. M. Soft Robotics. *Angewandte Chemie - International Edition* **57**, 4258 (2018).
8. Rus, D. & Tolley, M. T. Design, fabrication and control of soft robots. *Nature* **521**, 467 (2015).
9. Shepherd, R. F., Ilievski, F., Choi, W., Morin, S. A., Stokes, A. A., Mazzeo, A. D., Chen, X., Wang, M. & Whitesides, G. M. Multigait soft robot. *Automotive Engineer (London)* **108**, 20400 (2011).
10. Tolley, M. T., Shepherd, R. F., Mosadegh, B., Galloway, K. C., Wehner, M., Karpelson, M., Wood, R. J. & Whitesides, G. M. A Resilient, Untethered Soft Robot. *Soft Robotics* **1**, 213 (2014).
11. Kim, S., Laschi, C. & Trimmer, B. Soft robotics: A bioinspired evolution in robotics. *Trends in Biotechnology* **31**, 287 (2013).
12. Deimel, R. & Brock, O. A Novel Type of Compliant, Underactuated Robotic Hand for Dexterous Grasping. *Proceedings of Robotics: Science and Systems* (2014).
13. Trivedi, D., Dienno, D. & Rahn, C. D. Optimal, model-based design of soft robotic manipulators. *Journal of Mechanical Design* **130**, 91402 (2008).
14. Ilievski, F., Mazzeo, A. D., Shepherd, R. F., Chen, X. & Whitesides, G. M. Soft robotics for chemists. *Angewandte Chemie - International Edition* **50**, 1890 (2011).
15. SoftGripping. *SoftGripping.com*

16. Hajash, K., Sparrman, B., Guberan, C., Laucks, J. & Tibbits, S. Large-scale rapid liquid printing. *3D Printing and Additive Manufacturing* **4**, 123 (2017).
17. Seibel, A. & Schiller, L. Systematic engineering design helps creating new soft machines. *Robotics and Biomimetics* **5** (2018).
18. Wehner, M., Truby, R. L., Fitzgerald, D. J., Mosadegh, B., Whitesides, G. M., Lewis, J. A. & Wood, R. J. An integrated design and fabrication strategy for entirely soft, autonomous robots. *Nature Publishing Group* **536**, 451 (2016).
19. Lin, H. T., Leisk, G. G. & Trimmer, B. GoQBot: A caterpillar-inspired soft-bodied rolling robot. *Bioinspiration and Biomimetics* **6** (2011).
20. Seok, S, Onal, C. D., Cho, K, Wood, R. J., Rus, D & Kim, S. Meshworm: A Peristaltic Soft Robot With Antagonistic Nickel Titanium Coil Actuators. *IEEE/ASME Transactions on Mechatronics* **18**, 1485 (2013).
21. Mao, S., Dong, E., Jin, H., Xu, M. & Low, K. H. Locomotion and Gait Analysis of Multi-limb Soft Robots Driven by Smart Actuators. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2438 (2016).
22. Onal, C. D. & Rus, D. Autonomous undulatory serpentine locomotion utilizing body dynamics of a fluidic soft robot. *Bioinspiration & biomimetics* **8**, 026003 (2013).
23. Tolley, M. T., Shepherd, R. F., Karpelson, M., Bartlett, N. W., Galloway, K. C., Wehner, M., Nunes, R., Whitesides, G. M. & Wood, R. J. An untethered jumping soft robot. *IEEE International Conference on Intelligent Robots and Systems*, 561 (2014).
24. Chakrabarti, A, Shea, K, Stone, R, Cagan, J, Campbell, M, Hernandez, N. V. & Wood, K. L. Computer-Based Design Synthesis Research: An Overview. *Journal of Computing and Information Science in Engineering* **11** (2011).
25. Cheney, N., Bongard, J. C. & Lipson, H. Evolving Soft Robots in Tight Spaces.
26. Cheney, N., Clune, J. & Lipson, H. *Evolved Electrophysiological Soft Robots* in *Artificial Life Conference Proceedings 14* **14** (2013), 222.
27. Cheney, N., MacCurdy, R., Clune, J. & Lipson, H. Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding. *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation - GECCO '13*, 167 (2013).
28. Rieffel, J., Knox, D., Smith, S. & Trimmer, B. Growing and Evolving Soft Robots. *Artificial Life* **20**, 143 (2013).
29. Cagan, J., Campbell, M. I., Finger, S. & Tomiyama, T. A Framework for Computational Design Synthesis: Model and Applications. *Journal of Computing and Information Science in Engineering* **5**, 171 (2005).
30. Trimmer, B. A., Lin, H.-T., Baryshyan, A., Leisk, G. G. & Kaplan, D. L. *Towards a biomorphic soft robot: design constraints and solutions* in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)* (2012), 599.

31. McMahan, W, Jones, B. A. & Walker, I. D. *Design and implementation of a multi-section continuum robot: Air-Octor* in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2005), 2578.
32. Laschi, C., Cianchetti, M., Mazzolai, B., Margheri, L., Follador, M. & Dario, P. Soft Robot Arm Inspired by the Octopus. *Advanced Robotics* **26**, 709 (2012).
33. Marchese, A. D., Onal, C. D. & Rus, D. Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators. *Soft Robotics* **1**, 75 (2014).
34. Zhong, Y. & Du, R. Design and implementation of a novel robot fish with active and compliant propulsion mechanism. *Robotics: Science and Systems* **12** (2016).
35. Bartlett, N. W., Tolley, M. T., Overvelde, J. T. B., Weaver, J. C., Mosadegh, B., Bertoldi, K., Whitesides, G. M. & Wood, R. J. A 3D-printed, functionally graded soft robot powered by combustion. *Science* **349**, 161 (2015).
36. Coyle, S., Majidi, C., LeDuc, P. & Hsia, K. J. Bio-inspired soft robotics: Material selection, actuation, and design. *Extreme Mechanics Letters* **22**, 51 (2018).
37. Umedachi, T. & Trimmer, B. A. *Design of a 3D-printed soft robot with posture and steering control* in *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), 2874.
38. Vikas, V., Cohen, E., Grassi, R., Sözer, C. & Trimmer, B. Design and locomotion control of a soft robot using friction manipulation and motor–tendon actuation. *IEEE Transactions on Robotics* **32**, 949 (2016).
39. Du Pasquier, C., Chen, T., Tibbits, S. & Shea, K. Design and Computational Modeling of a 3D Printed Pneumatic Toolkit for Soft Robotics. *Soft Robotics* **6**, 657 (2019).
40. Hiller, J. & Lipson, H. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics* **28**, 457 (2012).
41. Onal, C. D., Chen, X., Whitesides, G. M. & Rus, D. Soft mobile robots with on-board chemical pressure generation. *International Symposium on Robotics Research (ISRR)*, 1 (2011).
42. Suzumori, K., Endo, S., Kanda, T., Kato, N. & Suzuki, H. A bending pneumatic rubber actuator realizing soft-bodied manta swimming robot. *Proceedings - IEEE International Conference on Robotics and Automation*, 4975 (2007).
43. Kriegman, S., Walker, S., Shah, D., Levin, M., Kramer-Bottiglio, R. & Bongard, J. Automated shapeshifting for function recovery in damaged robots. *arXiv preprint arXiv:1905.09264* (2019).
44. Ishige, M., Umedachi, T., Taniguchi, T. & Kawahara, Y. Exploring Behaviors of Caterpillar-Like Soft Robots with a Central Pattern Generator-Based Controller and Reinforcement Learning. *Soft Robotics* **00**, 1 (2019).

45. Rieffel, J., Rife, J., Trimmer, B. & Saunders, F. *Evolving Soft Robotic Locomotion in PhysX* in *Proceedings of the 11th annual conference companion on genetic and evolutionary computation conference: Late breaking papers* (2009), 2499.
46. Andrew Spielberg, Allan Zhao, Tao Du, Yuanming Hu, Daniela Rus, W. M. *Learning-In-The-Loop Optimization : End-To-End Control And Co-Design of Soft Robots Through Learned Deep Latent Representations* in *Conference on Neural Information Processing Systems (NeurIPS* (2019).
47. Sims, K. Evolving virtual creatures. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1994* **SIGGRAPH '**, 15 (1994).
48. Lipson, H & Pollack, J. B. Automatic design and manufacture of robotic lifeforms. *Nature* **406**, 974 (2000).
49. Lund, H., Hallam, J. & Lee, W.-P.L.W.-P. Evolving robot morphology. *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)* (1997).
50. Stump, G. M., Miller, S. W., Yukish, M. A., Simpson, T. W. & Tucker, C. Spatial Grammar-Based Recurrent Neural Network for Design Form and Behavior Optimization. *Journal of Mechanical Design* **141**, 1 (2019).
51. Cheney, N., MacCurdy, R., Clune, J. & Lipson, H. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution* **7**, 11 (2014).
52. Hiller, J. D. & Lipson, H. Evolving Amorphous Robots. *Artificial Life XII. Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems*, 717 (2010).
53. Stanley, K. O. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines* **8**, 131 (2007).
54. Krishnamurti, R. & Stouffs, R. Spatial grammars: motivation, comparison, and new results. *CAAD Futures*, 57 (1993).
55. Stiny, G. Introduction to shape and shape grammars. *Environment and planning B* **7**, 343 (1980).
56. Hsiao, S.-W. & Chen, C.-H. A semantic and shape grammar based approach for product design. *Design Studies* **18**, 275 (1997).
57. Pugliese, M. J. & Cagan, J. Capturing a rebel: modeling the Harley-Davidson brand through a motorcycle shape grammar. *Research in Engineering Design* **13**, 139 (2002).
58. Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P. & Paragios, N. Shape Parsing via Reinforcement Learning. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 1 (2011).
59. McKay, A., Chase, S., Shea, K. & Chau, H. H. Spatial grammar implementation: From theory to useable software. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **26**, 143 (2012).

60. Shea, K. *Exploration in Using an Aperiodic Spatial Tiling as a Design Generator* in *Design Computing and Cognition'04* (2004), 137.

61. Shea, K. & Cagan, J. Innovative dome design: Applying geodesic patterns with shape annealing. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing* **11(05)**, p. 379 (1997).

62. Zimmermann, L., Chen, T., Shea, K., Zimmermann, L., Chen, T. & Shea, K. *Generative Shape Design using 3D Spatial Grammars , Simulation and Optimization* in *Design Computing and Cognition'16* (2017), 297.

63. Hiller, J. & Lipson, H. Dynamic simulation of soft heterogeneous objects. *arXiv preprint arXiv:1212.2845* (2012).

64. Sofa. *Sofa*

65. Couman, E. *Bullet Physics Library* 2017.

66. Hu, Y., Liu, J., Spielberg, A., Tenenbaum, J. B., Freeman, W. T., Wu, J., Rus, D. & Matusik, W. ChainQueen: A real-time differentiable physical simulator for soft robotics. *Proceedings - IEEE International Conference on Robotics and Automation* **2019-May**, 6265 (2019).

67. 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. *Artificial Life* **23**, 206 (2017).

68. Brodbeck, L., Hauser, S. & Iida, F. Morphological evolution of physical robots through model-free phenotype development. *PloS one* **10**, e0128444 (2015).

69. Königseder, C. & Shea, K. Visualizing Relations Between Grammar Rules, Objectives, and Search Space Exploration in Grammar-Based Computational Design Synthesis. *Journal of Mechanical Design* **138**, 101101 (2016).

70. Sutton, R & Barto, A. *Reinforcement Learning – An Introduction* 2e, 550 (2018).

71. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv preprint arXiv:1801.01290* (2018).

72. Ziebart, B. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy* PhD thesis (2010), 1.

73. Moseley, P., Florez, J. M., Sonar, H. A., Agarwal, G., Curtin, W. & Paik, J. Modeling, Design, and Development of Soft Pneumatic Actuators with Finite Element Method. *Advanced Engineering Materials* (2016).

74. Dreyer, M. *Simulation and Fabrication of Soft Actuators* PhD thesis (2016).

75. Hiller, J. D. & Lipson, H. Multi material topological optimization of structures and mechanisms. *11th Annual Conference on Genetic and Evolutionary Computation*, 1521 (2009).

76. Möller, M., Heidelberger, B., Hennix, M. & Ratcliff, J. Position based dynamics. *Journal of Visual Communication and Image Representation* **18**, 109 (2007).
77. Bender, J., Koschier, D., Charrier, P. & Weber, D. Position-based simulation of continuous materials. *Computers and Graphics (Pergamon)* **44**, 1 (2014).
78. Bender, J., Müller, M. & Macklin, M. Tutorial: Position-Based Simulation Methods in Computer Graphics. *Eurographics (Tutorials)* (2015).
79. Macklin, M., Müller, M., Chentanez, N. & Kim, T.-Y. Unified Particle Physics for Real-Time Applications. *ACM Trans. Graph. Article* **33** (2014).
80. Bender, J. *Position Based Dynamics Libarary* 2015.
81. Tricarico, M. *Design of a shape memory alloy activated soft robot* PhD thesis (2017).
82. Van Diepen, M. & Shea, K. A Spatial Grammar Method for the Computational Design Synthesis of Virtual Soft Robots. *Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1 (2018).
83. Van Diepen, M. & Shea, K. A Spatial Grammar Method for the Computational Design Synthesis of Virtual Soft Locomotion Robots. *Journal of Mechanical Design* **141**, 101402 (2019).
84. Triki, E., Collette, Y. & Siarry, P. A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. *European Journal of Operational Research* **166**, 77 (2005).
85. Calisti, M., Picardi, G. & Laschi, C. Fundamentals of soft robot locomotion. *Journal of The Royal Society Interface* **14** (2017).
86. Haarnoja, T. *Github Soft Actor-Critic* 2018.
87. Robert, Shepherd, F. *Soft Robot Walking and Crawling* 2011.
88. Formlabs. *Form 2*
89. Stratasys. *uPrint*
90. Messin-Roizard, T. *Development and construction of an untethered pneumatic system for soft robots* Bachelor Thesis (ETH Zürich, 2018).
91. Arduino. *Arduino Uno*
92. Lee, J, Kim, W, Choi, W & Cho, K. Soft Robotic Blocks: Introducing SoBL, a Fast-Build Modularized Design Block. *IEEE Robotics Automation Magazine* **23**, 30 (2016).
93. Shah, J. J., Vargas-Hernandez, N. & Smith, S. M. Metrics for measuring ideation effectiveness. *Design Studies* **24**, 111 (2003).
94. Macklin, M., Müller, M. & Chentanez, N. XPBD : Position-Based Simulation of Compliant Constrained Dynamics. *Proceedings of the 9th International Conference on Motion in Games*, 49 (2016).

95. Mm Oller, T. & Trumbore, B. Fast, Minimum Storage Ray/Triangle Intersection. *Journal of graphics tools* **2**, 21 (1997).

96. Devillers, O. & Guigue, P. *Faster Triangle-Triangle Intersection Tests* tech. rep. (INRIA, 2002).

97. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*, 1 (2013).

98. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).

99. Hasselt, H. V., Guez, A. & Silver, D. Deep reinforcement learning with double q-learning. *Thirtieth AAAI conference on artificial intelligence*, 2094 (2016).

100. Zhang, F., Leitner, J., Milford, M., Upcroft, B. & Corke, P. Towards vision-based deep reinforcement learning for robotic motion control. *Australasian Conference on Robotics and Automation, ACRA* (2015).

101. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 1 (2014).

102. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. & Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

103. Haarnoja, T., Hartikainen, K., Abbeel, P. & Levine, S. Latent Space Policies for Hierarchical Reinforcement Learning. *arXiv preprint arXiv:1804.02808* (2018).

CURRICULUM VITAE

PERSONAL DATA

Merel Dirkje Marie van Diepen

Date of Birth	June 23, 1983
Place of Birth	Heemskerk, Netherlands
Citizen of	Netherlands
Contact	merelvandiepen@gmail.com

EDUCATION

2011	ETH Zürich, Switzerland *Final degree:* MSc. BioMechanical Engineering at Delft University of Technology
2010	ETH Zürich, Switzerland *Final degree:* BSc. Mechanical Engineering at Delft University of Technology

EMPLOYMENT

2014 - 2020	Research and Teaching Assistant to Prof. Dr. Kristina Shea *Engineering Design and Computing Laboratory, ETH Zürich, Switzerland*
2012 - 2014	Motion Control Engineer *Ampelmann, Delft, Neteherlands*
2011 - 2012	Researcher Information and Communication Technologies *Almende, Rotterdam, Netherlands*
2008	Intern Remote Handling and Maintenance ITER *FOM - Foundation for Fundamental Research on Matter, Rijnhuizen, Netherlands*

PUBLICATIONS

ARTICLES IN PEER-REVIEWED JOURNALS:

1. Van Diepen, M. & Shea, K. A Spatial Grammar Method for the Computational Design Synthesis of Virtual Soft Locomotion Robots. *Journal of Mechanical Design* **141**, 101402 (2019).

To be submitted:

2. Van Diepen, M. & Shea, K. To be submitted: A Computational Design Synethesis of Physical Soft Robots. *Journal of mechanical design* (2020).
3. Van Diepen, M. & Shea, K. To be submitted: Co-Design of Morphology and Actuation of Soft Robots for Locomotion. *Journal of mechanical design* (2020).

CONFERENCE CONTRIBUTIONS:

Conference paper:

1. Van Diepen, M. & Shea, K. A Spatial Grammar Method for the Computational Design Synthesis of Virtual Soft Robots. *Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1 (2018).

Presentations:

2. Van Diepen, M. & Shea, K. *Presentation: Computational Design Synthesis of Soft Robots* in *ACM SIGCHI Summer School on Computational Fabrication and Smart Matter* (2017).
3. Van Diepen, M. & Shea, K. *Presentation: Computational Design Synthesis of Virtual Locomotive Soft Robots* in *Fields Workshop on Robust Geometric Algorithms for Computational Fabrication* (2019).

Poster presentation:

4. Van Diepen, M. & Shea, K. *Poster Presentation: Computational Design Synthesis of Virtual Soft Robots* in *8th Intl Conf on Design Computing and Cognition / DCC'18* (ed Gero, J. S.) (2018).