Czech Technical University in Prague

Faculty of Electrical Engineering

Bachelor thesis

Motion planning for modular robots under failures

Author:
Daniel Lamper

Supervisor:
Ing. Vojtěch Vonásek

Department of Cybernetics

Prague

May, 2016

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Daniel L a m p e r

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Motion Planning for Modular Robots under Failures

Guidelines:

1. Get familiar with modular robotics [3], with focus to locomotion generation [4,5]. Study physical simulation of 3D objects [9]. Study bio-inspired optimization methods like Genetic Algorithms or Swarm-based systems [6,7].
2. Implement a simple simulation model of modular platform CoSMO [8]. Implement an optimization method suitable for optimization of locomotion patterns.
3. Optimize locomotion patterns for several modular robots (e.g. 'cross' robot, snake-like, caterpillar) using the optimization method implemented in task 2.
4. Implement a motion planning method (e.g. RRT [1,2]) that uses the optimized locomotion patterns.
5. Analyze influence of joint failures. Consider two types of failures (free-joint and stuck-joint). Design rules to recover from failures (e.g. using detaching of modules, re-learning of patterns, etc.)
6. Verify designed strategies for failure recovery in simulation.

Bibliography/Sources:

[1] LaValle, Steven M - Planning algorithms - Cambridge university press, 2006.
[2] LaValle, Steven M., and James J. Kuffner Jr. - Rapidly-exploring random trees: Progress and prospects - (2000).
[3] P. Levi, E. Meister AC. van Rossum, T. Krajnik, V. Vonasek, P. Stepan, W. Liu, F. Caparrelli - "A cognitive architecture for modular and self-reconfigurable robots" - Systems Conference (SysCon), 2014 8th Annual IEEE, vol., no., pp.465,472, March 31 2014-April 3 2014
[4] Q. Wu et al. - "Survey of locomotion control of legged robots inspired by biological concept." - Science in China Series F: Information Sciences 52.10 (2009): 1715-1729
[5] A. J. Ijspeert - "Central pattern generators for locomotion control in animals and robots: a review." – Neural Networks 21.4 (2008): 642-653.
[6] R. Eberhart, J. Kennedy. - "A new optimizer using particle swarm theory" - Proceedings of the sixth international symposium on micro machine and human science. Vol. 1. 19
[7] R. Eberhart, Y. Shi, J. Kennedy – "Swarm Intelligence" - Morgan Kaufmann, 2001
[8] Liedke, J. – "The Collective Self-reconfigurable Modular Organism (CoSMO) " - IEEE/ASME International Conference on Advanced Intelligent Mechatronics - 2013
[9] David H. Eberly - Game physics - CRC Press, 2010.

Bachelor Project Supervisor: Ing. Vojtěch Vonásek

Valid until: the end of the summer semester of academic year 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, December 20, 2015

Author statement for undergraduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date.......................... ....

signature

Abstract

In this thesis, methods of motion planning for modular robots under failure is investigated. Methods based on Central Pattern Generator (CPG) are used to generate signals for locomotion control of modules. The Particle Swarm Optimization (PSO) is bio-inspired method which is used for optimization of CPG-based locomotion. For purpose of the motion planning, the Rapidly-exploring Random Tree (RRT) algorithm is implemented. In this thesis, two types of module failure are investigated, fix-joint and free-joint. The influence of different combinations of modules under failures was statistically compared on different types of robots. The experiments were conducted using a simulation environment. The result of the experiments is that in most of the cases robots operate better with broken modules as without them.

Keywords: failure, modular, locomotion, planning, robot

Abstrakt

V tejto práci sú skúmané metódy plánovania pohybu pre modulárne roboty s uvažovanou poruchou modulov. Metódy založené na Central Pattern Chenerator (CPG) boly použitý pre generovanie signálov pre pohyb . Particle Swarm Optimization (PSO) algoritmus je metóda inšpirovaná prírodou a bola použitá pre optimalizáciu pohybu založenom na CPG. Pre účely plánovania pohybu bol implementovaný Rapidly-exploring Random Tree (RRT) algoritmus. V tejto práci sú skúmané dva typy poruchy modulu, zaseknutie kĺbu a neovládateľnosť kĺbu. Vplyv rôznych kombinácií pokazených modulov bol štatisticky porovnaný na rôznych typoch robotov. Experimenty boli vykonané za použitia simulačného prostredia. Výsledkom pokusov je, že vo väčšine prípadov roboty pracujú lepšie s pokazenými modulmi ako bez nich.

Klúčové slová: porucha, modulárny, pohyb, plánovanie, robot

Acknowledgements

I would like to thank my supervisor, Ing. Vojtěch Vonásek Ph.D., for the patient guidance, a lot of useful advices and his time. Finally, I would like to thank my family for their support.

Contents

Chapter 1

Introduction

The modular robotics is a field of study of the robotic systems which are constructed from small bodies called "modules". These modules are independent mainly with their own sensors, actuators, computed means and docking interface. This architecture allows a different types of structures of modular robots. These structures are referred to as constructions in this thesis, in the Fig (1.2) is an example of construction of CoSMO robot. This variability makes modular robots very flexible, adaptable and makes economical advantage in contrast with the fixed robotics structures. Their flexibility and adaptability is based on the ability of reconfiguration or to self-reconfiguration of modular robotics systems. In the Fig (1.1) an example of modular robot's ability to reconfigure construction is shown. The modules in the robot are docked to each other by a docking interface which is being used for transfer forces, torques, electrical power and communication between modules.

On the other hand, there are several issues connected to modular robotics. Such as increasing degrees of freedom (DOF) with the growing number of modules, what make gaits of a modular robot more complicated to control. In task of modular robotics it is not suitable to use Inverse Dynamic or Kinematic Problem for solving the motion of robot. There exist several methods how to undergo this issue.

In this thesis, for generating locomotion control the Central Pattern Generator (CPG) is used. The CPG is optimized by an Particle swarm optimization (PSO) algorithm. For motion planning Rapidly exploring Random Tree (RRT) algorithm is used. These algorithms are described in detail in Chapter (2) & Chapter (3).

The locomotion in modular robotics means the ability of robot to move by generating gaits such as walking, side-winding, rolling, crawling, undulating and others. The locomotion applied in specific direction is called primitive. The modular system can produce as many primitives as possible. However, in this thesis robots considering of most four primitives.

The thesis introduces modular robotics under failure and studies several options how to approach this issue. There exist two kinds of failure in modular robotics: electrical and

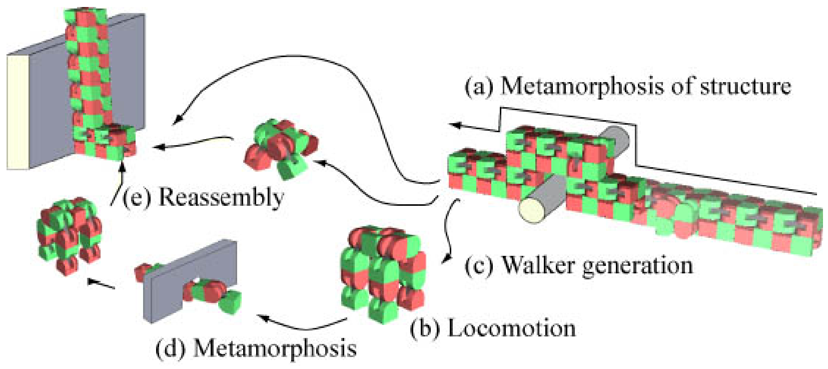


FIGURE 1.1: Adaptability of M-tran modular robot.[1]

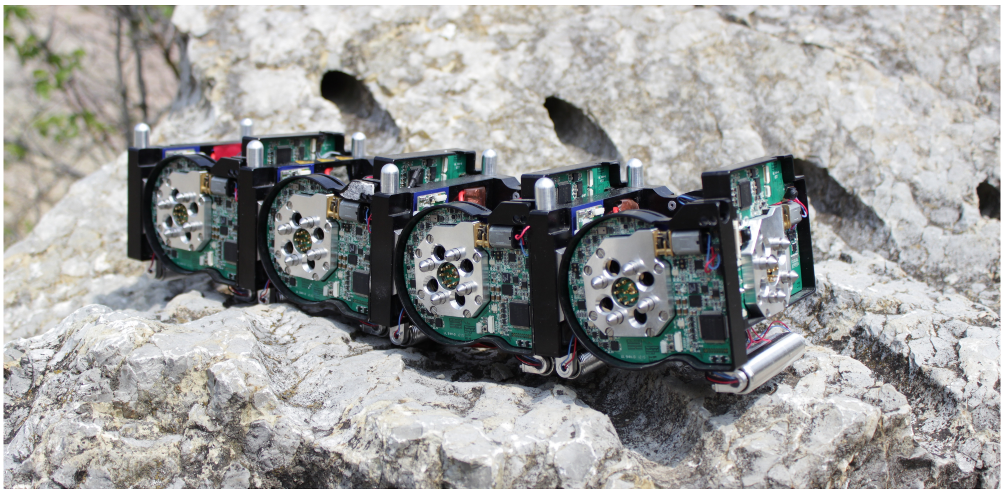


FIGURE 1.2: Example of Snake-Like construction of CoSMO modular robot.

mechanical. For purpose of this thesis, there are considered only mechanical types of failure. The Chapter (4) examines an influence of failure on robot's behavior in environment, its locomotion ability and capability to search space. The study does not investigate the cause of failure but handles an option how to overcome this issue and fix the problem. The thesis considers two types of mechanical failures: fix-joint and free-joint. Fix-joint failure means that the module gets stuck in a last position, while the free-joint failure makes a rotation of the hinge uncontrollable and free to revolve.

Chapter 2

State of the art

This thesis deals with motion planning for modular robots as well as modular robots under failures. This is achieved by using the concept of motion primitives. The low-level locomotion is generated using CPG (2.2). The motion planning is realized using RRT (2.4.1). The state-of-the-art methods relevant to the studied topics are briefly described in this chapter.

2.1 Modular robotics

Modular robot is a robotic system made by a set of connected units (modules) which are joined together by a docking interface.[2] Modules can be described as simple, self-contained building blocks with their own sensors, actuators and computed means. Due to the ability of the modules to join, they can build up specific robotic structures. Modular robots can adapt to the specific environment which means, they are able to reconfigure or to self-reconfigure[3;4]. The reconfiguration can be achieved e.g. using external tools (of other robots). Contrary, self-reconfiguration is such a reconfiguration that can be performed by the robot itself. The robots with self-reconfigurable abilities are therefor more complex. For example, a robot can change the structure from legged robot to a snake-like robot which is suitable when the terrain change. In case of failure of one or more modules, modular robots have ability to repair / self-repair. This fact makes modular robots more economically profitable over fixed structure robots.

2.1.1 Base Model

Modular robots are classified into two major categories: the *Mobile **C**onfiguration **C**hange* (MCC) and the ***W**hole **B**ody **L**ocomotion* (WBL).[2]

Mobile Configuration Change: One of the main characteristics of MCC category is that individual robots' modules interact with the environment independently. Modules are physically connected to one another and make head-to-tail form. The locomotion patterns are traditional and use typical mechanism that enables mobility of individual modules such as wheels or triads. MCC contains means of coupling what enable a swarm of modules and brings possibility to build large configurations. Examples of modular robots from the MCC category:

- *S-BOT*[8] — each module is autonomous and operate with the mobile tracked-and-wheeled platform, the arm and the gripper.
- *Uni-Rover*[9] — has been developed as a planetary rover. Rover's wheels were transfered to self-contained mobile modules referred to as a child. This child can connect to/ disconnect from mother platform. The child is made by gripper mechanism and caster wheel.

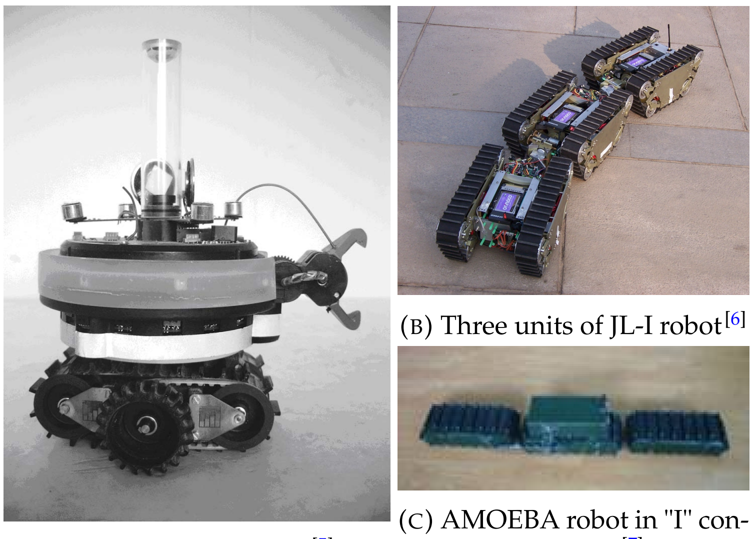

(A) Module of S-BOT[5]
(B) Three units of JL-I robot[6]
(C) AMOEBA robot in "I" configuration[7]

FIGURE 2.1: Examples of modular robots of the MCC category

- *JL-I[10]* — contains cone-shaped connector with matching coupler in center of module and two sides tracked units.
- *Millibots[11]* — two male steel pins in front of module coupled with female receptacle in back of module.
- *AMOEBA[12]* — Modules of this robot can be joined from all sides.

Whole Body Locomotion: Unlike MCC category, WBL category's modular robots are more flexible for movement and can provide various gaits such as walking, side-winding, rolling, crawling, undulating and others. However, modular robots from the WBL category can provide useful movement only under the condition that they are connected to the structure.[13] The WBL category is divided by architecture to three subcategories[2]: **a)** chain architecture **b)** lattice architecture and **c)** hybrid chain-lattice architecture.

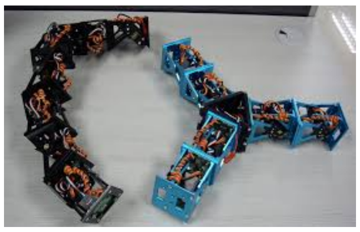
a) Chain architecture This architecture provides whole body locomotion which involves many degrees of freedom (DOF). Legged and snake-like structures are typical for the chain modular robots. Examples of chain modular robots are depicted in Fig (2.3).

- *PolyBot[17]* — It is self-reconfigurable modular robot providing multiple whole body locomotion.
- *GZ-I[18;15]* — This type is similar to PolyBot, but without any self-reconfigurable abilities.
- *CKBot[19]* — It can be docked manually or can provide self-reconfiguration by using permanent magnets. Three out of many locomotions he provides are walking, undulating and rolling.

b) Lattice architecture Characteristic for this architecture is the grid position of modules which are connected via docking interface. The lattice architecture contains three different possible mechanisms: *Macro robots*, *Mini robots* and *Transferable mechanism*[2].

- *Macro robots* — Mostly have cubic or parallelepipedic modules.

(A) CKBot[14]


(B) Snake-like robot and three-leg robot configurations of GZ-I modular robot[15]

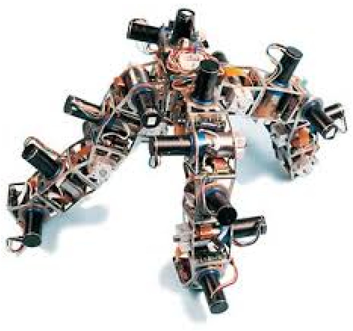
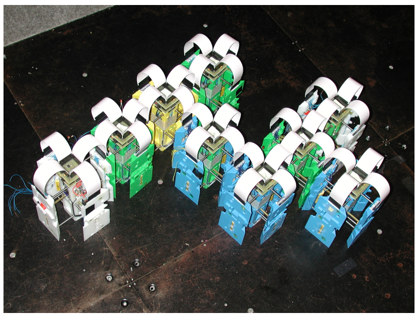

(C) PolyBot[16]

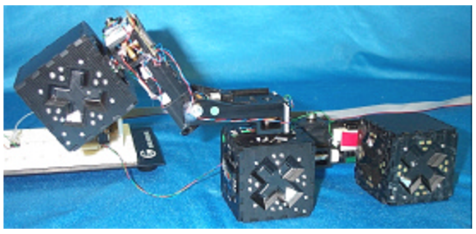
FIGURE 2.2: Examples of modular robots of WBL in a chain architecture

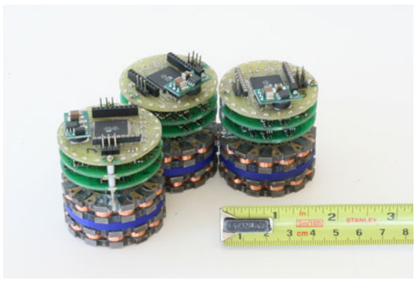
- *Mini robots* — Interesting on mini modular robots are potential military and civilian applications. These applications include exploring in environment unsuitable for humans or large robots. It is technical challenge to mini-size reconfigurable robotics, especially battery size.
- *Transferable mechanisms* — Mechanisms comprises of multiple magnetic gears where each one contains motor providing rotation around the central axis. On each gear unit there is used a multi-pole magnet with a total of six poles (3N, 3S).

c) Hybrid chain-lattice architecture This architecture has been investigated for mobile reconfigurable furniture application as well as for other highly adaptive mobile systems.

- *M-TRAN[1]* — This hybrid-morphology robot consists from semi-cylindrical modules which are composed of one passive and one active semi-cylinder.
- *ATRON[20]* — Robot with near-spherical and pyramidal modules. Each module is constituted of two four-sided pyramids with edge carvings that enable the module exhibit a near spherical geometry.
- *SuperBot[21]* — Modules of SuperBot own structure with two half-cubes rotating around a central axis and enable the module to achieve pivot around three different links.


(A) Crystalline robot[22]


(C) iCubes modular robot[24]


(B) Catons[23]

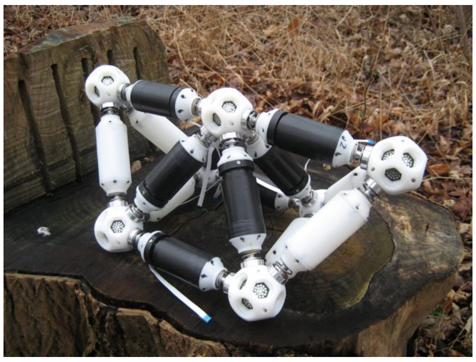

(D) Odin robot[25]

FIGURE 2.3: Examples of modular robots of WBL in a chain architecture

2.1.2 CoSMO

Another example of the hybrid architecture is CoSMO which is described in detail below.

The Collective Self-Reconfigurable Modular Organism[26] (CoSMO) is example of swarm robots. CoSMO is type of the modular robot of hybrid architecture that consists from several building blocks (modules). Modules are cubes of size 105x105x105 mm and weight of 1,25 kg and they encompass docking devices that are being used for transfer of forces, torques, electrical power and communication between modules. The CoSMO modules can dock between each other from every of four sides.

2D locomotion of CoSMO robots is provided by 2D-drive units based on two Archimedes Screws. The screw-drives are used for different kind of movements. Depending on rotation of the screw-drives the CoSMO can move in different ways. If screw-drives rotate in same direction, the CoSMO will move forward or backward, on the other hand if screw-drives rotate in different direction the CoSMO will move sideways. For CoSMO rotation, one of the screw-drives has to be controlled and the other one needs to be motionless.

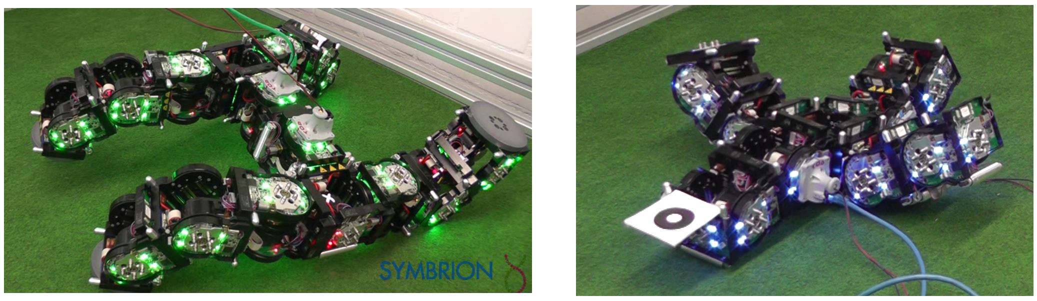


FIGURE 2.4: Lizard (left) and Cross (right) modular robots made of CoSMO modules.

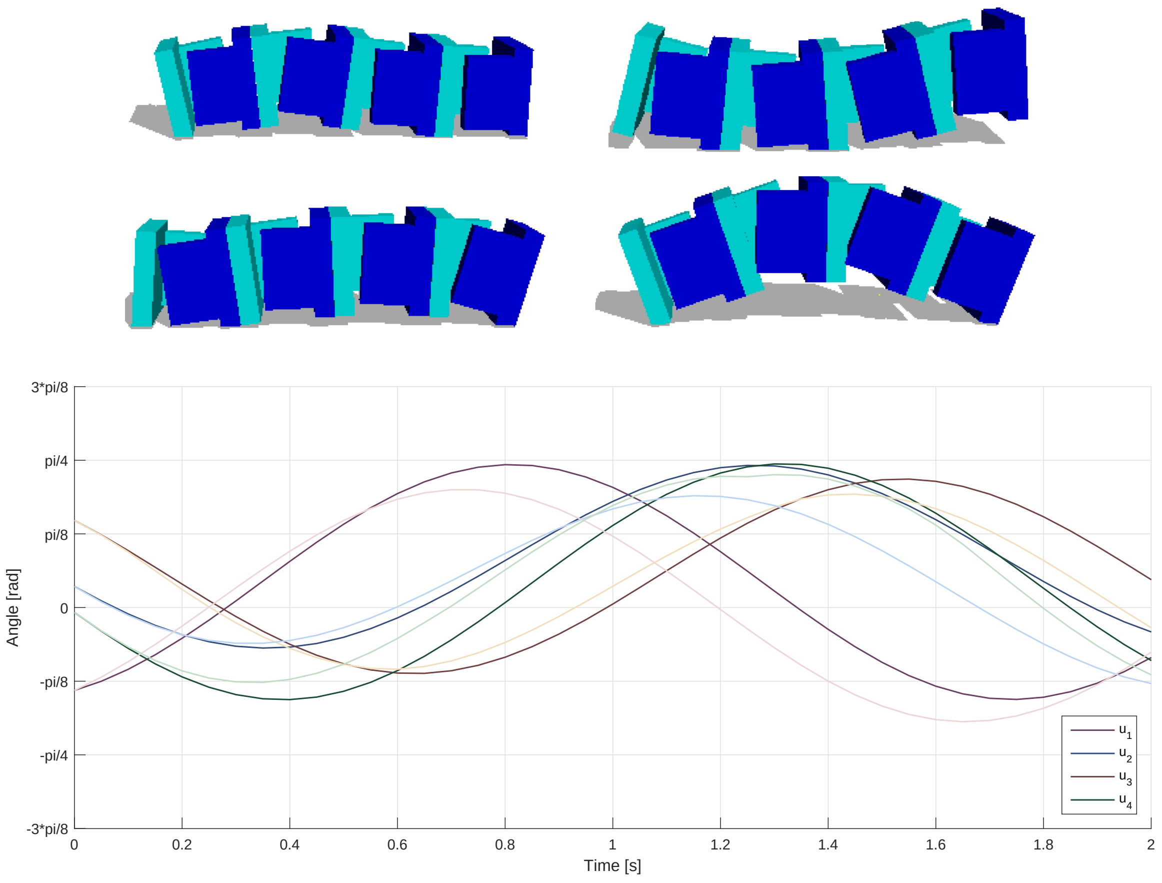


FIGURE 2.5: A snake-like robot driven by Eq. (2.1). The modules are numbered from right to left.

For 3D locomotion of CoSMO robots is used the main hinge, which can move in range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ rad. The hinge is controlled by setting a desired angle which is generally controlled by CPG (Sec 2.2). However, in this thesis the angle is given by the angle speed Eg. (2.1), where $A_i \wedge B_i \in \langle -\frac{\pi}{4}, \frac{\pi}{4} \rangle$ rad, $\omega_i \in \langle 0.3, 0.7 \rangle$ Hz and $\varphi_i \in \langle -\pi, \pi \rangle$ rad. The joint can rotate very fast (the maximal hinge speed is $\frac{2}{3}\pi \; rad/s$), but the fast movements are not used in this thesis and the modules move very slowly.

$$u_i(t) \quad = \quad A_i \sin(\omega_i t + \varphi_i) + B_i \tag{2.1}$$

Inspiration

The design of CoSMO platform was inspired by the following scenario. In general, symbiosis is cooperation between numbers of different species. In robotic sphere the symbiosis means aggregate cooperation of particles (swarm robots). The article[27] discusses the example of aggregate cooperation of modular robots in the task of energy foraging. Modular robots (particles) are placed in unknown space. They mission is to stay alive and that is possible only if they find the charging station. In first out of three cases is the environment without any barriers, there are only particles (exactly in the article is mentioned 70 robots). For finding the charging station, these particles cooperate only by sharing the information about they position and status of finding the charging station. Only few robots can be charged at a same time. So after finding the charging station, there is created a queue of robots which wait for their time of charging. In second case, in front of the charging station is added the barrier and the mission of particles is same, to stay alive. In this situation all robots die because no one found the energy source. In last case there is still barrier in environment. However, particles are taught higher cooperation as the aggregation of many single particles and they can be formed into the one multi-robot organism. That is the only way how they can pass the barrier and stay alive.

2.1.3 Failures in modular robotics

As was mentioned before, modular robots have abilities to repair /self-repair when they are under failure. That makes an important advantage of the modular robots architecture. The easiest solution how to fix the issue of broken module is to repair it with a spare module. However, this is possible only if the organism is near to a repair station, thus this solution is not possible if the robot operates in the field. When the robot is on mission, the solution can be to disconnect the broken module. It might seem as easy way to tackle the problem, but it may not be true. The issue of this method may be inability to undock the module due to mechanical or electrical problems or the robot cannot self-reconfigure due to environmental conditions. When the robot disconnect the broken modules, his ability of locomotion can change. Because of the new construction robot's primitive can be unperformed. The article[28] investigates the adaptation of the robots' locomotion instead of exchanging of the broken modules. This thesis examines the robots' behavior in case of failure and study possible solutions to tackle the problem. It also investigates the difference between behavior of robots under failure and robots without the broken modules.

The number of possible failures N_m for a robot consisting of m modules is equal to:

$$N_m = \sum_{i=1}^{m} \binom{m}{i} \cdot k^i = (1+k)^m \,, \tag{2.2}$$

where k is a constant number of CPG-relevant failures, for example a discrete set of joint angles where the module might get stuck. In an example of robot in Fig (2.4) with $m = 16$ modules considering of most three non-working modules with failure level $k = 1$, this leads to:

$$N_{16} = \sum_{i=0}^{3} \binom{16}{i} \cdot 1^i = 697, \tag{2.3}$$

possible configuration of the robot. The number of possible failures grows exponentially with number of modules m or with number of defect types k. In this thesis the robot's construction contains $m = 9$ modules in maximum, considering of most four broken modules and the level of failure $k = 2$. So that makes 2851 possible functional states of the robot which need to be considered for optimization. This is lot of computation so just a little sample of them was investigated.

2.2 Central Pattern Generator

One of the main hallmark of robots with many DOF (e.g. legged or modular robots) is that they can easily step over barrier or go through uneven ground, what is great advantage in comparison with wheeled robots. Robots with many DOF have more flexible body, therefore they can deal better with more complex areas as wheeled robots. Unfortunately, some other issues, such as degree of freedom or balance keeping, make motion control of robots with many DOF more demanding and much complicated task. These issues are mainly solved by programming mechanism. Algorithms, that are used to cover these issues, are based on studying and simulating animals' walking mechanism. From engineering aspect, this is efficient way to overcome obstacles of motion control for robots with many DOF.

Engineers have developed and improved motion control method based on Central Pattern Generator[29] (CPG). CPG network is dynamic biological system which combines neural system, body and environment. The neural system produces periodic as well as non periodic control signals. Depending on the type of CPG, the signals can be generated with or without sensory feedback[30]. As the environment change through moving of body, the neural system adapt control signals as well. From engineering perspective, CPG can be treated as feedforward in addition with feedback control system. The CPG inspired control method has become a hot topic of study for engineering applications. There are several mathematical models commonly used in CPG related studies.

Example of Neuron CPG model is Hodgkin-Huxley (H-H) type model[29]. The original H-H model has many parameters so for better understanding of the basic behavior of neurons a simplified FitzHugh-Nagumo model can be defined:

$$\begin{aligned} \dot{x_i} &= c\left(y_i + x_i + \frac{x_i^3}{3} + f_{ci}\right), \\ \dot{y_i} &= -\frac{x_i - a + by_i}{c}, \end{aligned} \tag{2.4}$$

where x_i is the membrane potential of the i-th neuron; f_{ci} is the driving signal from neuron i and a, b and c are parameters.

Other example of Neuron CPG model is Stain's model[29], which is able to produce oscillations. It is defined by differential equations:

$$\begin{aligned} \dot{x_i} &= a\left(-x_i + \frac{1}{1 + \exp(-f_{ci} - by_i + bz_i)}\right), \\ \dot{y_i} &= x_i - py_i, \\ \dot{z_i} &= x_i - qz_i, \end{aligned} \tag{2.5}$$

where x_i is the membrane potential of the ith neuronal oscillator; f_{ci} is the driving signal for oscillator i; a is a constant affecting the frequency of the oscillation; b allows the model to adapt a change in stimulus; p and q control the rate of this adaptation.

The last mentioned Neuron CPG model is Leaky-integrator model[29], which describes basic behavior of neurons. It is true thatthis model is not able to simulate degree of adaptation of neurons however Matsouka proposed this model to better fit for the properties of neurons.

Other types of models are Nonlinear oscillator models. It is worth to mention Kuramoto's model, Hopf model, Van der Pol's model and Rezleigh's model.[29] Hopf model is represented by:

$$\begin{aligned}\dot{x} &= \left(\mu - r^2\right) x + \omega y, \\ \dot{y} &= \left(\mu - r^2\right) y + \omega x,\end{aligned} \tag{2.6}$$

where $\mu > 0$ is amplitude of output signal, $r = \sqrt{x^2 + y^2}$ and ω is angular velocity which control the frequency of the oscillator.

2.2.1 CPG in Modular Robotics

CPGs have been originally studied for legged robots, however they can also be used to control locomotion of modular robots. Modular robots are made up from identical building blocks (modules), which can work independently and alone or can cooperate with a number of other modules in many different configurations. One of the examples of modular robots using the CPG model for gait control is Roombot[31]. The CPG are used due to their ability to generate periodic oscillations which is suitable Roombot locomotion. The specific model used for Roombots is the Hopf model, defined by Eq. (2.6). The Roombot module is created by four oscillators where three of them are used for locomotion and last one represents a clock. The goal is to investigate the control method which comes true by following these implementations criteria:

- *Morphology Independent* — Strategy planning is independent to morphology of robots.
- *Life-long Learning* — Algorithm for strategy planning should not be designed for particular morphology because through the time robots can change his form because of environment changes or some module can fail.
- *Noise Tolerance* — Strategy must be tolerant to noisy fitness measurements.
- *Simple Implementation* — If strategy is to be morphologically independent then it must be able to work at each module. Modules are embedded devices, that means they have limited communication and computation abilities so implementation must require a minimal amount of resources.

For optimization of CPG patterns was selected Simultaneous Perturbation Stochastic Approximation[31] (SPSA) method. This method is independent on the number of modules as well as morphology of whole robots because SPSA optimize each module independently to other modules. Consequently SPSA is simple to implement in a distributed fashion.

2.3 Gait optimization

Behavior of CPGs can be influenced by their parameters. For example, the Hopf oscillate (E.g. 2.6) is controlled by parameters μ (amplitude of motion) and ω (speed). To realize a desired gait, the parameters need to be optimized according to defined cost function. Optimization of CPG parameters leads to the high-dimensional optimization, which can be solved e.g. using genetic algorithms.

2.3.1 Genetic Programming

Genetic programming[32] (GP) is offered as a efficient method for optimization. GP is a domain-independent and population of computer programs evolving approach to solve problems. GP's simulated evolution is based on the Darwinian principle of reproduction and survival of the fitness. The major attributes of GP are:

- *Function Set* — Since locomotion of Snakebots are based on periodical signals, there are included trigonometric functions sin and cos as well as algebraic functions $(+, -, *, /)$.
- *Terminal Set* — Terminal set includes terminal symbols necessary for finding-solution. Symbols of GP are represented as: `segment_ID` (unique index of segment of Snakebot), `specialize` (e.g. by phase, amplitude or frequency), `time`, `Pi` and `random constant` (range [0,2]).
- *Fitness evaluation* — Fitness function based on the velocity of Snakebot. The velocity values are typically within the range [0,2] and are multiplied by normalizing coefficient within the range [0,200].
- *Genetic representation* — Evolved genotypes of simulated Snakebots are represented as trees.
- *Genetic operations* — Binary tournament selection is used because of its robust, commonly used selection mechanism and because is simple to code. Crossover operations are defined by sub-trees as the way of the same data type which can be loose only from parents.

GP evaluation and CPG-based (2.2) approaches for locomotion gaits share same features, such as the open-loop control scheme and the incorporation of coupled oscillators. However for CPG-based approaches is necessary to have at least little domain-specific knowledge about the task, what on the other hand GP does not need to know all domain-specific limitations and in spite of this it can give optional solutions.

Example of application of GP in modular robotics

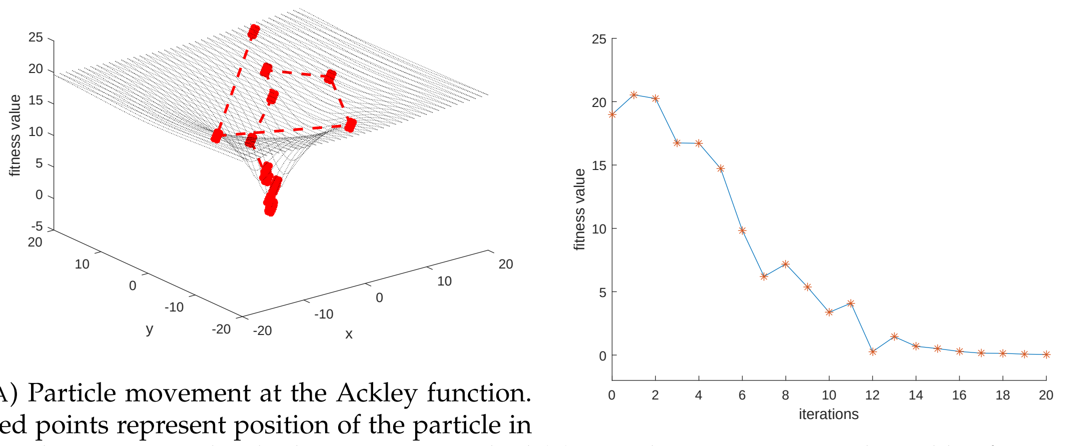
The article[33] discusses snake-like robots (Snakebots) and how GP is used for investigating the fastest possible locomotion. Snakebots are limbless, wheelless robots using sidewinding locomotion. The advantages of Snakebots are their robust characteristics and ability to traverse terrain that can cause troubles for wheeled or legged robots. The Snakebots may be cheaper and easier to build, in case we have separate modules already constructed. Because of their simple construction it is easy to replace module in case of failure. Other useful features are for instance good traction, high redundancy and stability. On the other side in comparison with legged or wheeled robots, Snakebots are more difficult on thermal control and have smaller payload. In principle, Snakebots may be formulated by a mathematical model, however this model would be defined with lot of variable and equations what makes this task really hard to describe and understand. Considerable amount of degrees of freedom, which depends on each other, makes this task complex and hard to describe via easy understandable mathematical model. Ability of GP to find a near-optional solution in a reasonable runtime is offered as effective method how to go through this task.

2.3.2 Particle swarm optimization

Particle swarm optimization[34;35] (PSO) algorithm is an optimization algorithm based on the inspiration from birds' flock. The swarm contains particles where each has its own position $\vec{x}_i$ in n-dimensional space. These particles are moved around to examine the search-space and to find relative best solution with the possible best fitness value. Except actual position and fitness value the particles also remember local best known position $\vec{p}_i$. The swarm at all has global best known position $\vec{p}_q$ where index q is pointer to global best position. An important part of the PSO algorithm is the velocity $\vec{v}_i$ of each particle, which is defined by[36]:

$$\vec{v}_i \quad = \omega \vec{v}_i + \varphi_p r_p (\vec{p}_i - \vec{x}_i) + \varphi_q r_q (\vec{p}_q - \vec{x}_i), \tag{2.7}$$

where φ_p, φ_q and ω are parameters which control the behavior and efficacy of the particles' movement in search-space and r_p and r_q are random real numbers in range [0, 1]. The velocity is changing each time step for each particle i toward p_i and p_g position.



(A) Particle movement at the Ackley function. Red points represent position of the particle in search-space in individual iteration. Dashed line can be imagine as way which traversed the particle

(B) Particle convergence at the Ackley function.

FIGURE 2.6: Example of using PSO for investigating the minimum of the Ackley function Eq. (2.8)

$$F(x) \quad = 20 + \exp(1) - 20 \exp\left(-0.2\sqrt{\frac{1}{32}\sum_{i=1}^{32} x_i^2}\right) - \exp\left(\frac{1}{32}\sum_{i=1}^{32}\cos(2\pi x_i)\right) \tag{2.8}$$

2.4 Motion planning

The task of motion planning is to find a feasible trajectory from a given initial configuration to the goal configuration. The trajectory is feasible if the robot does not collide with any obstacles. Motion planning[37;38] can be defined as a computation of geometrical primitives. The motion planning algorithms are used for finding a path in n-dimensional search-space. Also these algorithms might have included a collision detection module. If a solution exists, the algorithm must return one in finite time. However, if the solution does not exist the algorithm may run forever. To avoid this possibility the motion planning algorithms include a bound, e.g. number of iteration while the algorithm should find the solution is set. Unfortunately, this may cause that a solution would not be found in spite of that it might exist. So it is necessary to consider number of iterations that are optimal to be used.

There exist two different types of planning: the *Path planning* and the *Trajectory/Motion planning*. The path planning produces a geometric graph. The output of this method is a simple vector containing only geometric coordinates, x, y and in 3D space also z coordinates of vertexes in space, while the motion planning method considers kinematics as well as

Algorithm 1: Basic PSO algorithm

```
Input  : Space bounds $\vec{B}$; Velocity bounds $\vec{V}$; Fitness function $F$
Output: Best position $p_q$
1  $q = 0$;                  // $q$ is index of best global position
   // initialize coordinates of particles
2  for $i$ = 1 to number of particles do
3      $\vec{x}_i$ = rand($\vec{B}$);
4      $\vec{p}_i = \vec{x}_i$;
5      if F ($\vec{p}_q$)<F ($\vec{p}_i$) then
6          $q = i$;
7      end
8      $\vec{v}_i$ = rand($\vec{V}$)
9  end
10 for $k = 1$ to Number of iterations do
11     for $i = 1$ to number of particles do
12         for $d = 1$ to dimension do
13             $v_{i,d} = v_{i,d} + \varphi_p$rand(0, 1) $(p_{i,d}$-$x_{i,d})$+$\varphi_q$rand(0, 1) $(p_{q,d}$-$x_{i,d})$;
14             if $v_{i,d} > V_d$ then
15                 $v_{i,d} = V_d$;
16             end
17             if $v_{i,d} < -V_d$ then
18                 $v_{i,d} = -V_d$;
19             end
20             $x_{i,d}$ += $v_{i,d}$;
21         end
22         if F ($\vec{p}_i$)<F ($\vec{x}_i$) then
23             $\vec{p}_i = \vec{x}_i$;
24             if F ($\vec{p}_q$)<F ($\vec{p}_i$) then
25                 $q = i$;
26             end
27         end
28     end
29 end
```

dynamics of the robot. The results of motion planning is a trajectory describing how to robot move, i.e. it contains control inputs to the actuators. For purpose of this thesis the motion planning was used instead of path planning.

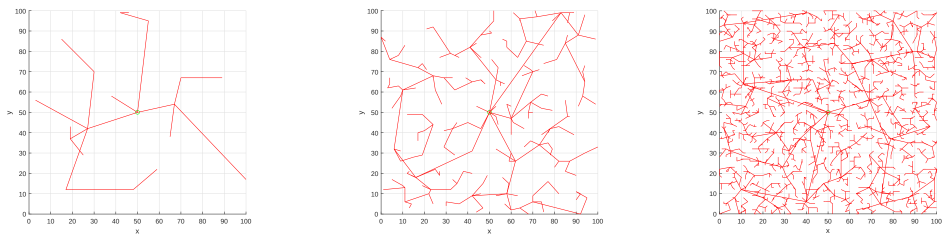
The motion planning algorithms are usually based on random sampling of approaches, which means that the found solution is not an optimal solution. Thus, motion planing can generally be described as a search for the goal configuration $\vec{q}_{goal}$ in n-dimensional space S. Each point $\vec{q} \in S$ is specified by its configuration attributes as the position, the orientation and so on. Most common motion planning algorithms are Probabilistic Roadmap algorithm (PRM)[39], Expansive Space Trees algorithm (EST)[40], Fast Marching Tree (FMT)[41], Bi-Directional FMT (BFMT)[42]. In this thesis, Rapidly-exploring Random Tree algorithm (RRT)[43;44]is used for motion planing. The RRT algorithm is described in Sec (2.4.1).

2.4.1 Rapidly-exploring Random Tree

Rapidly-exploring Random Tree (RRT) is the motion planning algorithm designed for finding a path in n-dimensional configuration spaces. In modular robotics it is used only for finding path in 2D or 3D space, so the thesis focused on these two types of planing task. As an input of the RRT algorithm is an initial configuration $\vec{q}_{init} \in S$, where S is a search space. In each iteration a random configuration $\vec{q}_{rand} \in S$ is chosen which will connect with its own nearest neighbor $\vec{q}_{near} \in G$. G is the graph of path planning. The new random configuration is added to graph and this step repeats until the goal configuration $\vec{q}_{rand}$ is found.

Algorithm 2: Basic RRT algorithm

```
Input  : Search-space $S$, Initial configuration $\vec{q}_{init} \in S$, Goal configuration $\vec{q}_{goal} \in S$
Output: RRT graph $G$
1  $G$.addVertex($\vec{q}_{init}$);
2  for $i = 1$ to $max\ iterations$ do
3      $\vec{q}_{rand} = S$.getRandomConfiguration();
4      $\vec{q}_{near} = G$.nearestNeighbor($\vec{q}_{rand}$);
5      $G$.addVertex($\vec{q}_{rand}$);
6      $G$.addEdge($\vec{q}_{rand}$, $\vec{q}_{near}$);
7      if $\vec{q}_{rand} = \vec{q}_{goal}$ then
8          return $G$;
9      end
10 end
```



(A) RRT with 20 iterations (B) RRT with 200 iterations (C) RRT with 2000 iterations

FIGURE 2.7: Examples of basic RRT algorithm with tree different numbers of iterations is same square space.

(A) RRT use in 3D cube bound space (B) RRT use in 3D bowl bound space

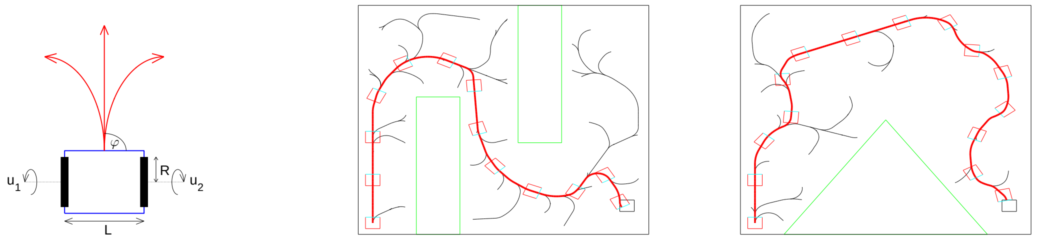
FIGURE 2.8: Examples of RRT motion planning in 3D space

Unfortunately, the basic algorithm predicts free space. That means space without any collision possibility. So collision detection had to be added into the algorithm. Collision detection can be realized e.g. by computing intersection between polygons.[45] Other important function of the RRT algorithm is the method for finding the nearest neighbor in an existing graph. For small graph could be used method which compares Euclidean distance between points in space. But for a large graph this method is unusable because of the time complexity $O(n)$. Due to this it is better use other more intelligent method. In this thesis, KD-tree[46] based algorithm is used for finding the nearest neighbor. The time complexity of searching in kd-tree is $O(\log n)$. The Fig (2.7) shows using basic RRT algorithm in 2D space with different number of iterations. The Fig (2.8) demonstrates the example of application in 3D space.

To get familiar with RRT motion planning, a simple RRT motion planing algorithm for differential wheeled robot was implemented. Because of its easy construction which is showed in Fig (2.9a) it was a good choice for learning basic ideas of motion planning in 2D environments. Also the differential robot can expand just into several directions. The differential robot moves in 2D space with velocity according to Eq (2.9), so the RRT algorithm saves position $\vec{q}$ and rotation φ of robot to graph **G**. The final condition is 300 iterations with size of space 100 x 100 or distance of robot to the goal position which was represented as a square with size which was equal to robot size. The size of the differential robot used in study is 5 x 5. Into the environment there were also added one or more obstacles represented by different polygons. Thus, the RRT algorithm contains a collision detection method which is based on the investigation of intersection between a point and a polygon. The important part of RRT algorithm for differential robot is the function for expanding of possible way which was added to the basic algorithm. The function which return the found path was added into the basic algorithm as well. For finding the nearest neighbor it is used the KD-tree based algorithm. The motion model of the differential wheeled robot is:

$$\begin{aligned}
\dot{x} &= \frac{R}{2}\cos\varphi\,(u_1 + u_2)\,, \\
\dot{y} &= \frac{R}{2}\sin\varphi\,(u_1 + u_2)\,, \\
\dot{\varphi} &= \frac{R}{L}\,(u_1 - u_2)\,,
\end{aligned} \tag{2.9}$$

where u_1 and u_2 represent the velocities of rotation of wheels, R is an radius of wheel, L is size of robot, φ is azimuth and x, y represent position in Cartesian coordinate system.



(A) Differential robot (B) Rectangle barriers (C) Triangle barrier

FIGURE 2.9: Examples of RRT motion planning for differential robot in 2D space

Chapter 3

Implementation

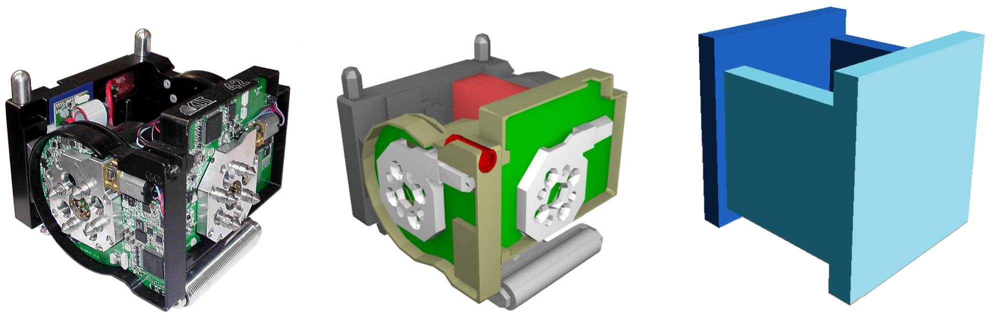
3.1 Simulation environment

For the purposes of this thesis, a simple simulator of CoSMO modular robots was created. Open Dynamic Engine (ODE)[47] was used for simulation of physical environment. The ODE also includes a library drawstuff that enables a simple visualization of the environment. The ODE contains a collision detection of geometric structures and allows connection between objects with joints of different types, e.g. hinge-joint, fix-joint, ball-joint etc. Moreover, it yields functions for applying torques and forces on geometrical objects.

The CoSMO module was modeled as a pair of L-shaped parts. These parts were connected by the hinge-joint and the resulting module is depicted in Fig (3.1). The constructions of robots, which are used in this thesis, are depicted in Fig (3.2) and Fig (3.6) . It is true that simplified model does not describe the hardware of real module, however it is sufficient for benchmark simulations and testing. The huge advantage of using the simplified model is faster simulation. Fix-joint is used in the simulated environment to imitate the real dock between modules. The simulator was designed for simulation of arbitrary shape of robots. The configuration of modules is written and loaded to ODE from a simple text file and by employing this data it is possible to create 3D shaped organism.

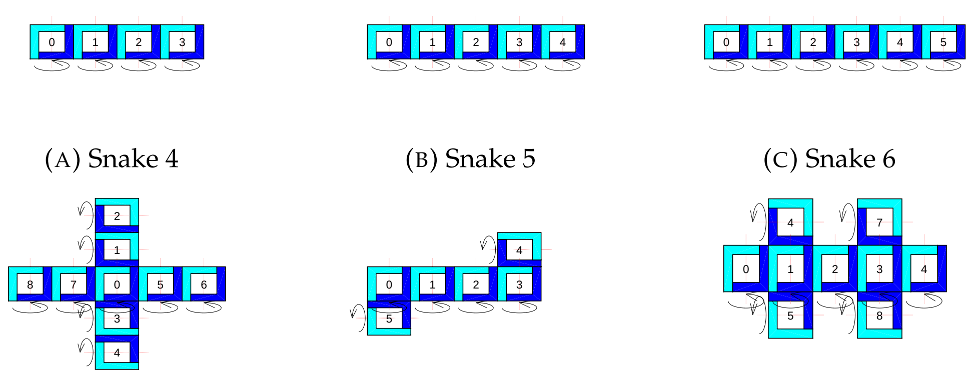
3.2 PSO optimization of CPG patterns for CoSMO robots

As was mentioned in (2.3.2), PSO algorithm is an optimization algorithm based on the swarm intelligence. In this thesis, PSO algorithm is applied for learning and optimizing of locomotion of the CoSMO robot. Each module of CoSMO robot is moved according to its velocity, which is defined by function (2.1). Variables of speed function are input for the



(A) The real module. (B) The detailed model. (C) The simple model used in simulator.

FIGURE 3.1: Modules of CoSMO robots used in the simulation.



(A) Snake 4 (B) Snake 5 (C) Snake 6

(D) Cross (E) S-shape (F) Dog

FIGURE 3.2: The CoSMO organisms used for experiments with the orientation of rotation axes. The arrows denote orientation of the rotation axis of each module joint.

optimization algorithm. Furthermore, in this thesis there are considered only four possible primitives: forward, backward, left and right, depending on which kind of CoSMO robot is used. For example snake like robot has only two possible primitives, forward and backward. The other employed constructions of CoSMO robot can performed all primitives.

The basic PSO algorithm has been upgraded in order to meet the criteria of optimization locomotion of the CoSMO robots. Initially, it is necessary to define the structure of $\vec{x}$ and $\vec{p}$ as well as structure of $\vec{v}$. The $\vec{x}$, $\vec{p}$ and $\vec{s}$ were not used as vectors but as two dimensional matrices **X**, **P** and **V** represented as:

$$\mathbf{X} = \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_m \end{pmatrix}, \ \mathbf{P} = \begin{pmatrix} \vec{p}_1 \\ \vec{p}_2 \\ \vdots \\ \vec{p}_m \end{pmatrix}, \ \mathbf{V} = \begin{pmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_m \end{pmatrix}, \tag{3.1}$$

$$\{\vec{x}_i, \vec{p}_i, \vec{v}_i\} = (A_1, \omega_1, \varphi_1, B_1, \ldots, A_n, \omega_n, \varphi_n, B_n), \tag{3.2}$$

where A_j, ω_j, φ_j and B_j are CPG parameters of each module of CoSMO construction. Therefore, $4n$ parameters need to be found to achieve a gait for modular robot with n modules.

3.2.1 Fitness evaluation

The fitness function measures quality of locomotion of a CoSMO robot. Higher result is better, because PSO searches for maximum of fitness function result. The fitness function depends on CPG parameters, construction of robot and the primitives of locomotion. The locomotion represents the test of ability to move the body in a primitive. The fitness function for the PSO algorithm is evaluated by the simulator. The PSO algorithm generates parameters of a CPG which determines the gait of the robot in simulator. The simulator runs for time T_{sim} with period t_{phys} which indicate accuracy of computation. For propose of this thesis $T_{sim} = 15$ s and $t_{phys} = 50$ ms. The evaluation of the fitness function is given by Eq (3.3) where x^{start} is the initial position of robot and x^{end} is the position of robot after locomotion. Index k in the equation represent a axis in which is locomotion related. And the variable a

can be ± 1, its value is depended on the direction of robot, if is direction in positive direction with axis the value of a is equal to $+1$, is is direction negative its value is equal to -1.

$$F(x^{start}, x^{end}, a) = \quad a(x_k^{end} - x_k^{start}), \tag{3.3}$$

Algorithm 3: Fitness function for PSO algorithm

```
input  : CPG parameters $\vec{x}$, CoSMO $robot$, Type of primitive $ToP$
output: Passed distance $dis$
1  $robot$.setParam($\vec{x}$);
2  $\vec{pos}^s$ = getPosition($robot$);
3  SimulateMove($robot$, 15);        // 15 as 15 seconds of simulation
4  $\vec{pos}^e$ = getPosition($robot$);
5  if $ToP = forward$ then
6      $dis = pos_x^e - pos_x^s$;
7  else
8      if $ToP = backward$ then
9          $dis = -(pos_x^e - pos_x^s)$;
10     else
11         if $ToP = left$ then
12             $dis = pos_y^e - pos_y^s$;
13         else
14             $dis = -(pos_y^e - pos_y^s)$;
15         end
16     end
17 end
18 $robot$.SetInitialConfig();
19 return $dis$
```

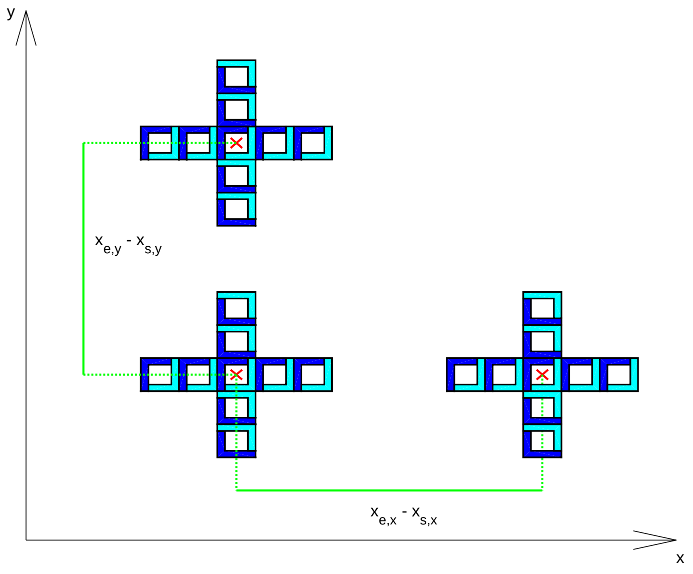
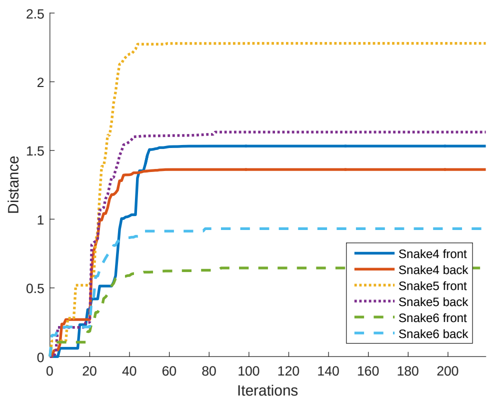
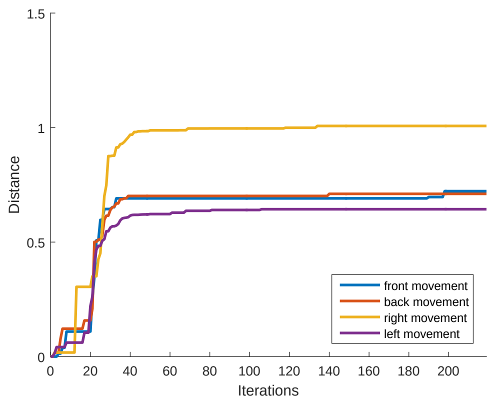


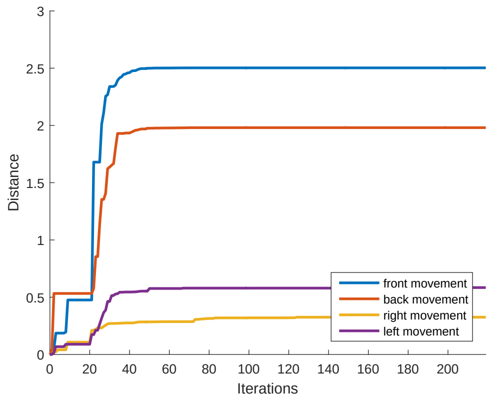
FIGURE 3.3: Example of the fitness evaluation.



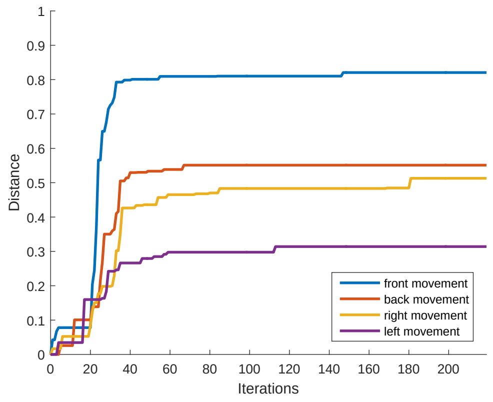
(A) Snake



(B) Cross



(C) S-sphere



(D) Dog

FIGURE 3.4: Convergence of fitness value of CPG pattern in PSO algorithm. The distance is measured in the units of module size.

3.3 RRT for motion planning of CoSMO robots

Employing the RRT algorithm for motion planning of CoSMO organism is more time consuming in contrast to the differential robot. The differential robot moves according to simple equation that can be solved analytically. Unfortunately, the locomotion of CoSMO organism is not so simple. It is not easy to derive an analytical function describing motion of general modular robots, as their motions depend both on the construction of modules as well as on the terrain. However, it is possible to use a simulation of locomotion to go through this task.

For purpose of this study the ODE[47] physical engine is used for simulation of physics and force computation in the simulation tool. The logic of algorithm is same as was in the case of the differential robot. The RRT algorithm generates a random coordinates of positions in 2D space. It is true that the CoSMO organism is able to move in an uneven 3D environment but in this thesis only empty flat space scene is used. However, the RRT algorithm has to remember for each vertex position $\vec{x} = \{x, y, z\}$, linear velocity $\dot{\vec{x}} = \{\dot{x}, \dot{y}, \dot{z}\}$, angular velocity $\dot{\vec{\varphi}} = \{\dot{\varphi}, \dot{\theta}, \dot{\psi}\}$ and quaternion $\vec{h} = \{h1, h2, h3, h4\}$. The vector of configuration looks as:

$$\vec{c} = \{\vec{M}_1, \vec{M}_2, \dots, \vec{M}_n\}, \tag{3.4}$$

Algorithm 4: Expand procedure

```
do
    $dis = MAX\_DOUBLE$;
    for $move := \{moveAhead, moveBack, moveLeft, moveRight\}$ do
        $\vec{q} = \vec{q}_{near} + move$;
        if IsNotInCollision($\vec{q}$) then
            if $|\vec{q}_{rand} - \vec{q}| < dis$ then
                $dis = |\vec{q}_{rand} - \vec{q}|$;
                $\vec{q}_{move} = \vec{q}$;
            end
        end
    end
    $G$.addVertex($\vec{q}_{move}$);
    $G$.addEdge($\vec{q}_{move}$, $\vec{q}_{near}$);
    $G$.addRoation($\vec{q}_{move}$);
    $\vec{q}_{near} = \vec{q}_{move}$;
while ($\vec{q}_{move} \approx \vec{q}_{rand}$);
```

where n represents number of modules M and ith module consists of:

$$\vec{M}_i = \{\vec{x}, \dot{\vec{x}}, \dot{\vec{\varphi}}, \vec{h}\}. \tag{3.5}$$

All of this configuration parameters have to be stored in RRT graph because of continuity of a locomotion, a trajectory and preserve numerical stability of simulation[48]. The output from RRT algorithm is a sequence of the motion primitives, i.e. array which contains list of numbers. Each of these numbers represent specific primitive which store information about the CPG's signals for each module of organism. These signals are read from text file and translated to the robot. For example, if the path from the initial position to the goal position includes n primitives, then the robot performs each primitive for $T_{sim} = 15$ s. After this time robot stops and stays with all its kinematic and dynamic properties. Then it is read the next primitive so there are ready CPG's signals of new primitive and the loop repeats until the robot comes to the goal position.

Nevertheless, this type of motion planning may cause several troubles. In the first step, the robot goes from the initial position where are all forces, torques, angular velocities and angles of hinges set to zero, or close to zero value. However, after first motion primitive the robot stops and maintains its dynamics. Thus, in next step the robot does not start from initial position and that means it does not pass the same distance as it could do so in the first step. From the experiments it was observed that robots can modify the direction within a primitive. Fortunately, this does not have big impact on searching space in many iterations, in spite of the fact that robot modifies direction of movement within a primitive, its ability to search space does not change. Because the robot modifies direction of locomotion in all primitives it still can go around whole space. It is true that it is backward gait but the robot does not care while it has constructions as were used in this thesis. The Fig (3.5) the robot is able to search whole space.

Behavior of the Cross, the Dog and the S-shape robot were similar. All of them found a way from initial position to goal position and also they were expand evenly in space. The

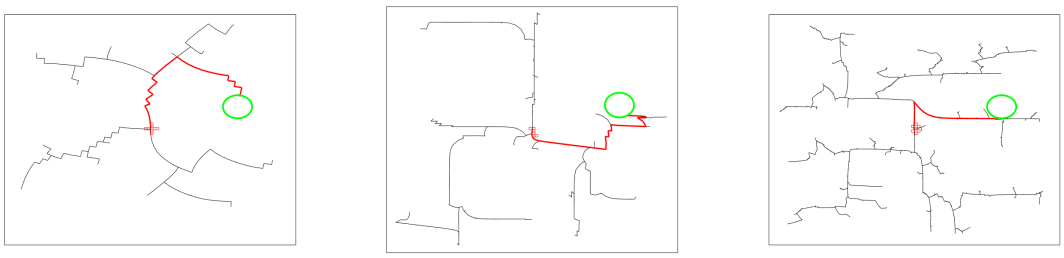

(A) Cross (B) S-shape (C) Dog

FIGURE 3.5: Example the motion planning of health CoSMO organisms.

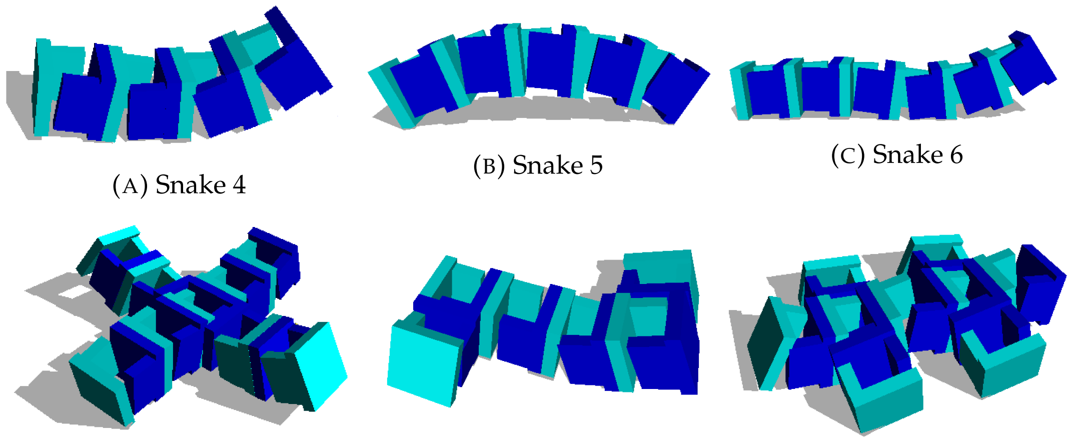

(A) Snake 4 (B) Snake 5 (C) Snake 6 (D) Cross (E) S-shape (F) Dog

FIGURE 3.6: 3D models of CoSMO robot in different constructions.

RRT tree of the Dog robot is bigger than others' but that is caused by a randomly set points in the search-space.

Chapter 4

CoSMO robot under failures

Every mechanical system has a tendency to fail. The CoSMO robot not being exception. There can be identified two types of failures: electrical and mechanical. In this study, there is described and investigated the mechanical type of failure. This section examines an effect of failures on behavior of robots, their locomotion ability and capability to explore an environment. The study does not investigate the cause of failure but handles an option how to overcome this issue and fix the problem. The thesis considers two types of mechanical failure: **a)** fix-joint and **b)** free-joint. Both failures make a module uncontrollable and affect the main hinge controlled by motor. The difference between fix-joint and free-joint failure is that the fix-joint failure sticks the hinge in a last position, so the module is stuck in state as is in the Fig (3.1). On the other hand the free-joint failure makes a rotation of the hinge absolutely free.

4.1 Implementation

The motion planning algorithm and PSO algorithm were used as in case of health robot. From the algorithm points of view, the difference between the health robot and robot under failure is in simulation part of algorithm. In this part the joint included in main hinge was changed. In case of fix-joint failure the joint of the main hinge was changed from the *hinge joint* to the *fix joint* what caused that the module cannot rotate and gets stuck in initial position. In case of the free-joint failure the forces and torques delivered from motor was deleted.

In the simulation, the robot is placed into the center of the area and RRT algorithm is run for a predefined amount of iterations, the number of iteration is equal to 300, with maximal 10 possible primitives per each iteration. Thus, the generated graph can include 3000 primitives in maximum. To measure influence of failures, there are proposed to measure area that is visited by the robot. To count this area, the space is discretized to a grid of cell size equal to 5 units, where one unit represents size of one module. The percentage of the visited area is then related to the locomotion performance.

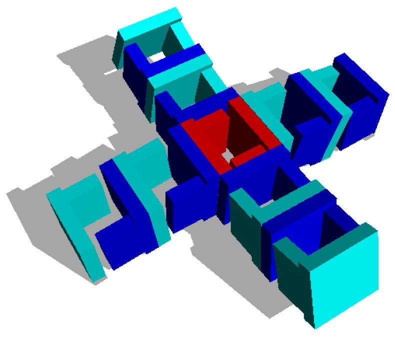
The output of RRT motion planning algorithm is graph G of searched space. After path planning, the percentage of visited area from the RRT's graph is calculated and if the percentage is equal or higher than threshold value for the specific construction of robot the algorithm is terminated with successful re-optimization. If the percentage is lower than threshold value the CPG patterns are optimized by the PSO and the RRT motion planning algorithm is run. This is repeated, while the percentage of the visited area is increased or until the threshold value is overcome. If the percentage of the visited area is decreased the broken modules are removed from the robot. Then the CPG patterns are optimized by PSO and the RRT algorithm is run. This again repeated, while the percentage of the visited area is increased or until the threshold value is overcome. If the percentage of visited area is decreased the message about unsuccessful re-optimization is returned. The pseudo-code is written in Alg (5).

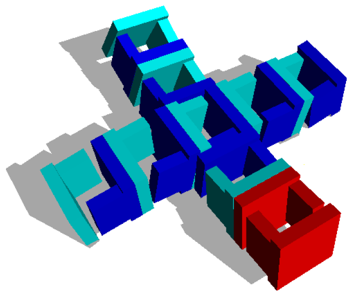
Algorithm 5: Motion planning algorithm with optimization for CoSMO robots under failure

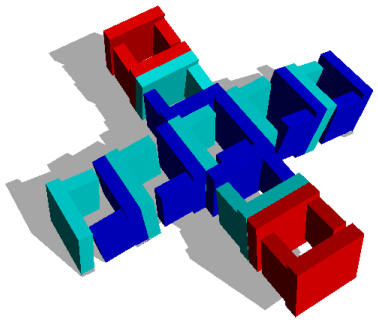
```
input  : Broken modules $\vec{bm}$, Robot $R$, Threshold value $ThresholdValue$
output: Percentage of visited area for each measurement, Result of re-optimization
$R$.CreateRobot($\vec{bm}$);
$G$ = calRRT($R$);
$p$ = getPercentil($G$);
if $p >= ThresholdValue$ then
    return Successful re-optimization
else
    callPSO($R$);
    return doOptimization($R, ThresholdValue, p, No$);
end

Procedure doOptimization
    input  : Robot $R$, Threshold value $ThresholdValue$, Last percentage
             $LastPercentage$, Are modules removed $areRemoved$
    output: Result of re-optimization
    $G$ = calRRT($R$);
    $p$ = getPercentil($G$);
    if $p >= ThresholdValue$ then
        if $areRemoved = No$ then
            return Successful re-optimization
        else
            return Modules were removed and re-optimization was successful
        end
    else
        if $p > LastPercentage$ then
            callPSO($R$);
            doOptimization($R, ThresholdValue, p, areRemoved$);
        else
            if $areRemoved = No$ then
                RemoveModul($R$);
                callPSO($R$);
                doOptimization($R, ThresholdValue, p, Yes$);
            else
                return Non-successful re-optimization
            end
        end
    end
```

4.2 Experiments

In figures below there are depicted the abilities to search space by different types of CoSMO robot under different failures. The magenta colored line represents the border and everything above this line is considered to be enough as a search of the scene. This border was created in pursuance of capability of health CoSMO robots to review the area. It is set little under the minimal measured value. For each robot under failure was simulated a RRT motion planning algorithm for ten times. Ten seems to be enough to create a credible view of behavior of the specific type of CoSMO construction. The CoSMO robot is self-reconfigurable so it should have capacity to disconnect a module which is out of order. The removing of these modules which are in center of the body, or which connect parts of the body, is represented as a substitute of these modules by working modules of body. That is possible only because of CoSMO robot's skill to self-repair and self-reconfigure.


(A) Cross_1 (fix-joint type of failure) & Cross_8 (free-joint type of failure)


(B) Cross_2 (fix-joint type of failure) & Cross_9 (free-joint type of failure)


(C) Cross_3 (fix-joint type of failure) & Cross_1 (free-joint type of failure)0

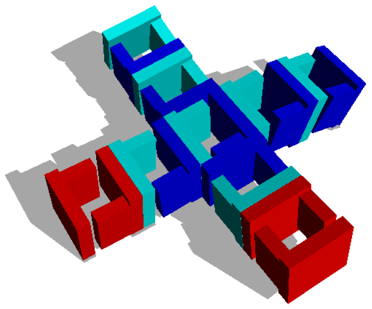

(D) Cross_5 (fix-joint type of failure) & Cross_1 (free-joint type of failure)2

FIGURE 4.1: The red colored modules are under failure.

4.2.1 Cross structure

The Cross type robot (3.6d) without failures visit 56.375% in average. With the exception of Cross_8 (4.1a), the Fig (4.2) displays measures of that kinds of CoSMO construction which were able to search more that 50% of the environment during every simulation. These robots have not removed any of modules, they just re-optimized their CPG patterns by PSO algorithm. The Cross_8 has average value under the threshold value, but it is in this figure because it has free-joined module in the center of body and this module cannot be removed. Therefore, generally it seems that failure of one or two modules which are not docked to

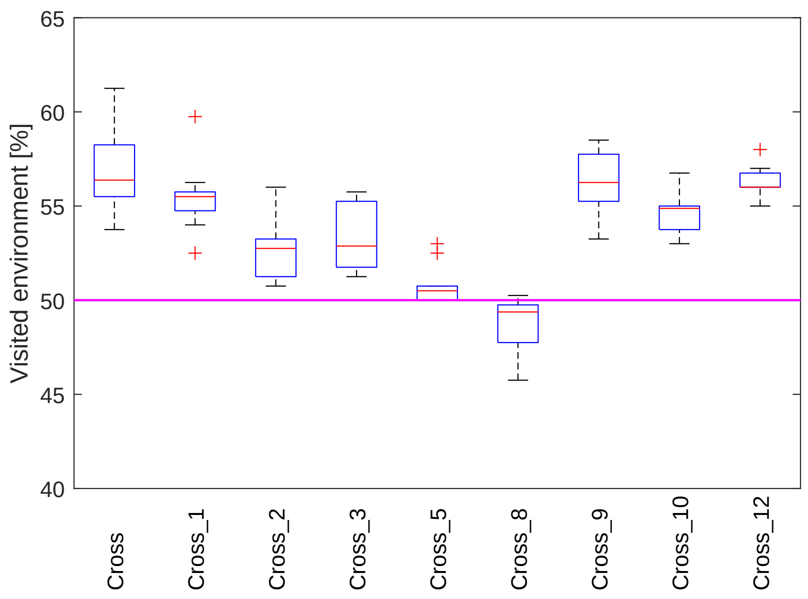


FIGURE 4.2: Figure shows the area that was searched by Cross robot. The area is represented in percentage. First box-plot depicts explored area by Cross robot which is not under failure.

each other, have just a little effect on the ability of locomotion of Cross construction CoSMO robot.

Though, not all of Cross type robots under failure are so adaptable. In the Fig (4.3) are drawn a box-plot graphs of these robots which were able to search less than 50% of the space. The cyan vertical line separates the simulation of locomotion before (*BD*) and after (*AD*) the removing of broken modules. The Cross type robot which has fix-jointed modules 1, 2 and 3 (Cross_6 (4.4b)) is on the threshold value before removing broken modules. However, the median value is under line so that is the reason why it is in this graph. Other robots under failure are little below the threshold. After removing a non-working modules of Cross_4 (4.5a), Cross_6 (4.5b) and Cross_7 (4.5c) robots, their ability to search space decreased. All this three robots have fix-joint malfunction of modules. So it seems that the Cross robot with fix-joint failure can better operate with broken modules as without them, even when they are docked to each other or the count of broken modules is higher than 2. On the other hand, in case of the free-joint failure the robots' ability to search space is not so accurate as in case of the fix-joint. These robots better operate without non-working modules as with them, especially Cross_11 (4.5a) and Cross_13 (4.5b).

The locomotion of the robot does not depend only on the robot's construction. It also depends on the CPG patterns. Really important part of the computation is PSO algorithm. For the optimization by PSO is used 200 iterations with 20 particles for each primitive. The design of Cross_4 and Cross_11 is same after removing of failed modules. Nevertheless, the difference between their percentage of visited space is obvious, what is caused by optimization of CPG patterns.

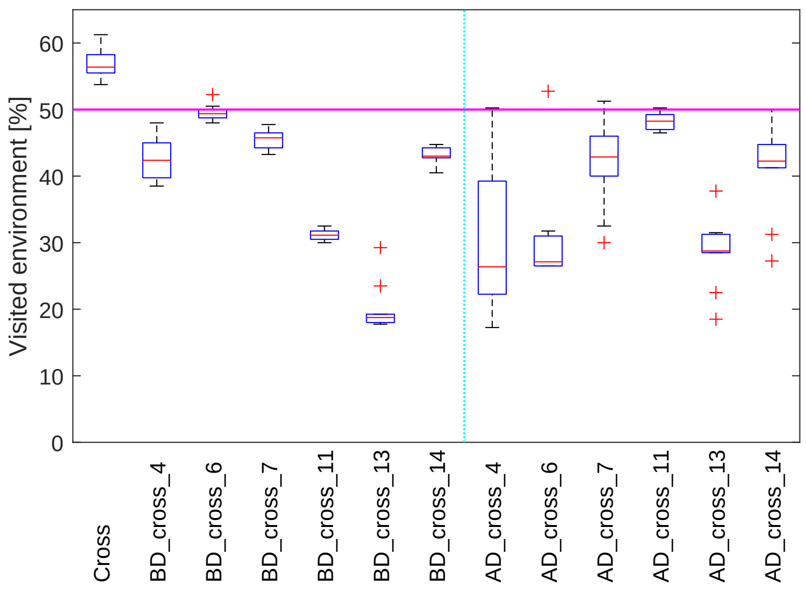
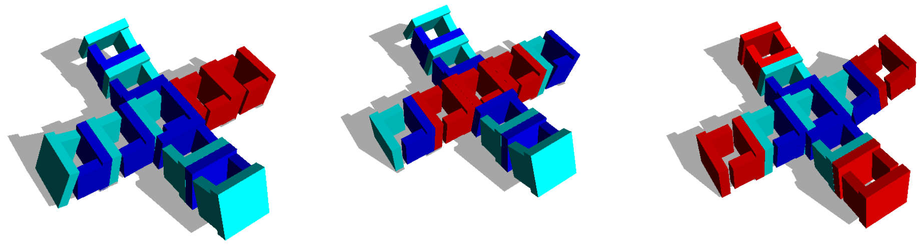


FIGURE 4.3: Figure shows a difference between the ability to search space of Cross type robots which operate with broken modules and robots which have removed the broken modules. Prefix BD means before removing modules so the robot is under failure and the prefix AD means that the robot has removed the modules.

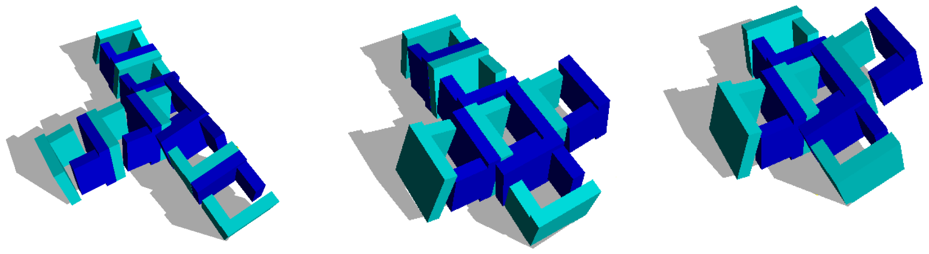


(A) Cross_4 (fix-joint type of failure) & Cross_12 (free-joint type of failure)

(B) Cross_6 (fix-joint type of failure) & Cross_13 (free-joint type of failure)

(C) Cross_7 (fix-joint type of failure) & Cross_14 (free-joint type of failure)

FIGURE 4.4: This pictures represent robots' construction with broken modules.



(A) Cross_4 & Cross_12 after removal of modules

(B) Cross_6 & Cross_13 after removal of modules

(C) Cross_7 & Cross_14 after removal of modules

FIGURE 4.5: This pictures show robots' construction after removal of broken modules

4.2.2 Dog structure

The next Fig (4.6) shows the percentage of visited area by the Dog designed CoSMO robot. It seems that ability of search the scene by Dog type robot under failure is not as good as it is in case of Cross robot. With the exception of the Dog_5 (4.7i), every other robot under failure was not pass the limit of searched space which is set to 30%. The median value of visited area by health Dog type robot (3.6f) is 36.875%. Probably the reason why is the Dog_5 so better than other robots is because it has broken head and tail modules, i.e. modules at the top and bottom of body. This robot's construction allows its to move without bigger problems and the stuck modules are used as a some kind of locomotion support. The three modules which are between head and tail are able to cope with the gait. The results for the Dog type robots cannot be generalize as it was done in case of Cross type robots.

It is true that Dog_1 (4.7a) has average value of searched space higher after removal of the modules than before but the variance is higher for this robot without broken modules. Also the maximal measured percentage is lower before than after removal. Thus, it cannot be said which variant is better for this robot. The next robot Dog_2 (4.7c) passed the threshold in tow measures for both construction, with and without broken modules. Nevertheless, the variance is lower and the median value is higher for the the robot under failures so from the statistic point of view is better for the robot operate under failures. The similar situation is in the case of robots Dog_7 (4.7d) and Dog_10 (4.7j). Dog_3 (4.7e) and Dog_6 (4.7b) they ability to search space with broken modules as well as without modules is not enough to pass the threshold value. The median value of both robots in both cases is almost equal to each other. Dog_4 (4.7g), Dog_8 (4.7f) as well as Dog_9 (4.7h) better operate without the broken modules as with them, thus for this robots is better to undock the broken modules.

As was mentioned before, the optimization by PSO algorithm plays main role in the computation and in overcoming the threshold value of percentage of visited area. In the Fig (4.6) as well as in the Fig (4.3) can be seen that robots which have same construction after removing the modules, do not visit the same percentage of area in average. This is

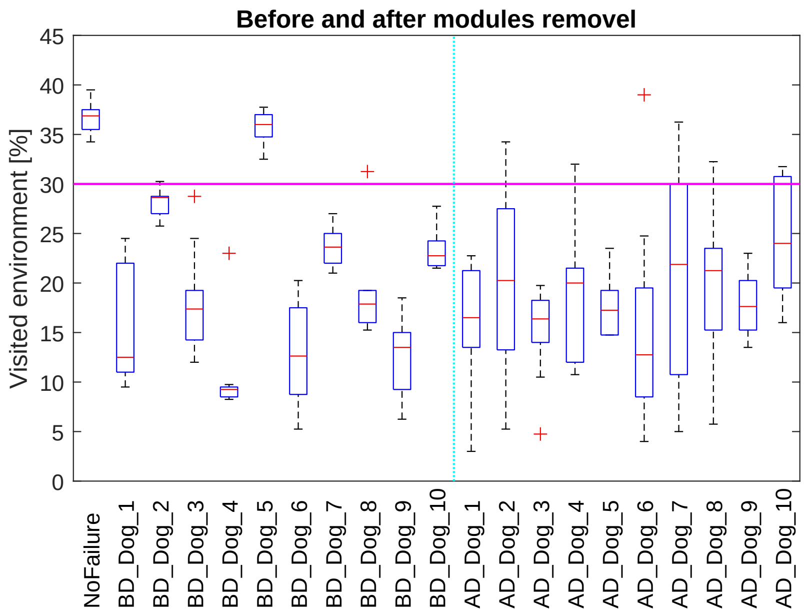
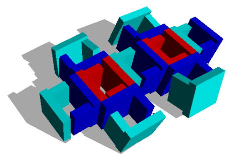
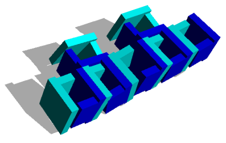
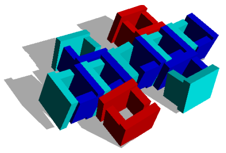
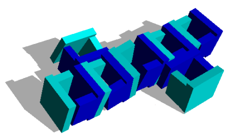


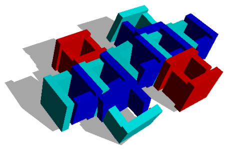
FIGURE 4.6: Figure shows a difference between the ability to search space of the dog type robots which operate with non-working modules and robots which have removed these broken modules. First box-plot represents searched space by healthy dog robot.

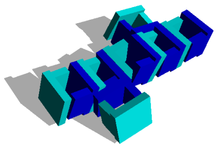

(A) Dog_1 & Dog_6.

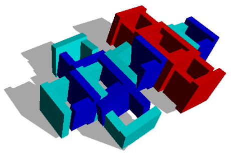

(B) Dog_1 & Dog_6 after removal of modules.

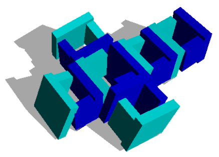

(C) Dog_2 & Dog_7.

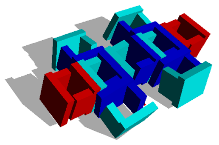

(D) Dog_2 & Dog_7 after removal of modules.


(E) Dog_3 & Dog_8.


(F) Dog_3 & Dog_8 after removal of modules.


(G) Dog_4 & Dog_9.


(H) Dog_4 & Dog_9 after removal of modules.


(I) Dog_5 & Dog_10.

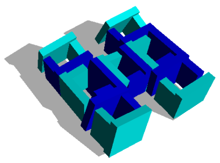

(J) Dog_5 & Dog_10 after removal of modules.

FIGURE 4.7: Dog construction of CoSMO robot. First name of Dog robot in captions of figures represents robot with fix-joint type failures, second name is for robots with free-joint failures.

caused by the PSO algorithm which optimizes a CPG patterns. The optimization does not start from zero fitness evaluation of the global best known configuration, but the global best known configuration is same as was in last optimization. Thus, for different types of failure the robot behavior is also different. Therefore, the convergence of CPG patterns is influenced by the previous computations and that is why there are different percentages of the visited area for same constructions.

4.2.3 S-shape structure

The last box-plot graph (4.8) demonstrates an computation of RRT planning algorithm for S-shaped type (3.6e) of CoSMO robot with couple of modules under failure. The first box-plot again represents a case of motion planing for S-shape construction with working modules. Other box-plots reflect search space by robots under failure. The locomotion of S-shape robot under failure is defective. However, this behavior could be expected for used cases of failures of modules. Modules on the bottom, in case of S-shape_1 (4.9a) as well as S-shape_3 (4.9b), and modules on the head, in case of S-shape_2 (4.9c) as well as S-shape_4 (4.9d), are uncontrollable so the robot ability to move in space is impaired. As can be seen in the Fig (4.10) the ability of locomotion to left and right side, decreased, especially movement into the right side. This is caused by the modules which are broken. It is interesting that in both cases, S-shape_1 has the path planning in Fig (4.10a) and S-shape_2 (4.10b), the robots do not move into the right side even though they do not have broken same modules. S-shape_1 has broken modules on the top of body and S-shape_2 has broken modules on the bottom of body, but both of them have fix-joint type of failure and that is probably reason why they cannot move into the right side. On the other hand S-shape_3 and S-shape_4 robots are able to move into the sides. In the Fig (4.10c) is shown the ability to search space by S-shape_3 robot and it seems that it is able to go into every side. But in more detailed observation, we should see that it does not move directly to the sides

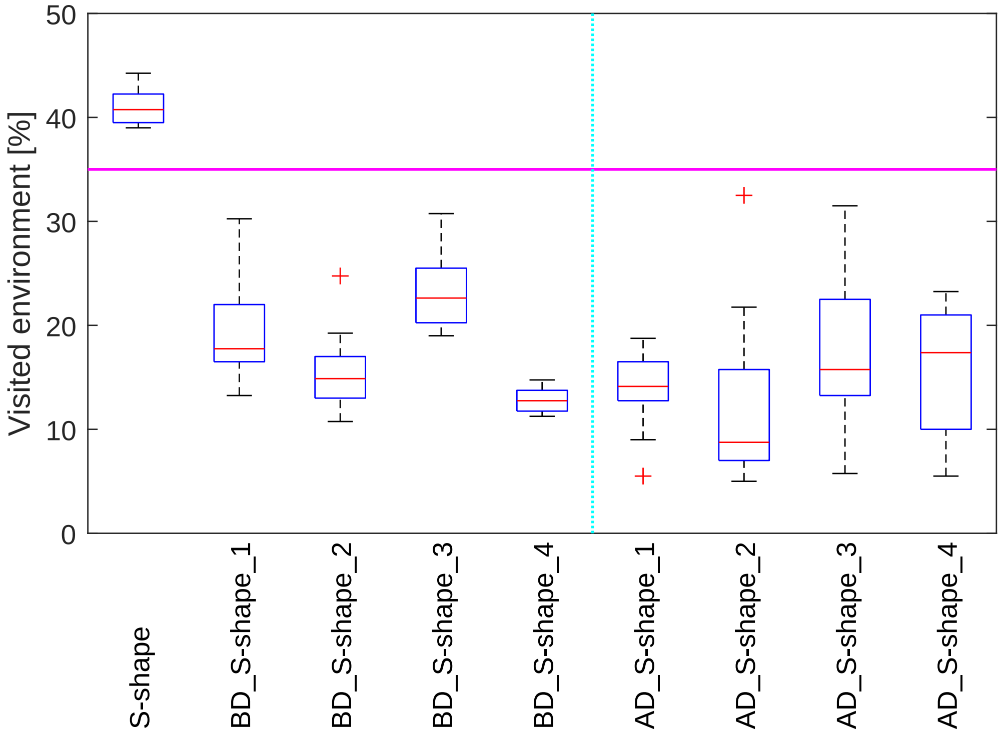
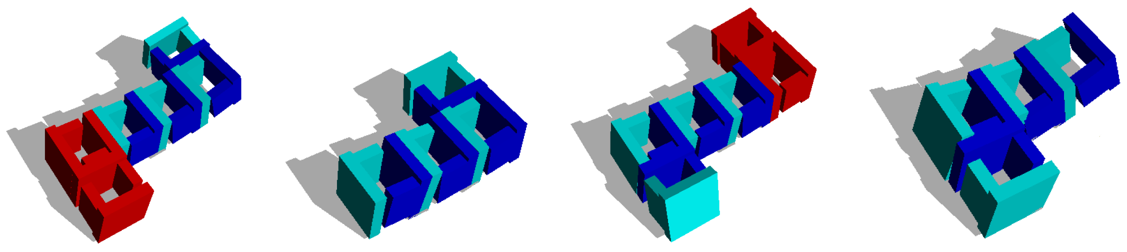


FIGURE 4.8: Figure shows a percentage of visited area for the S-shaped construction of CoSMO robot.

but it seems that it turned and then went ahead. Nevertheless, other robots were not able to visit as much space as the S-shape_3, but it does not matter how exactly robot does the gait or how it moves around the space, the important is to search space as much as possible.



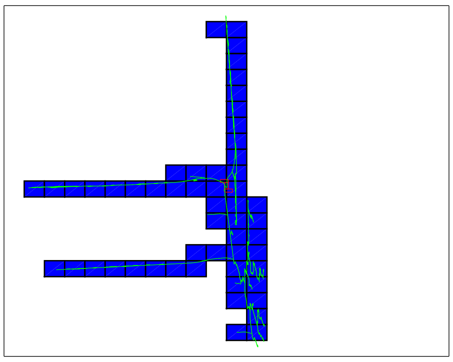
(A) S-shape_1 (fix-joint type of failure) & S-shape_3 (free-joint type of failure)

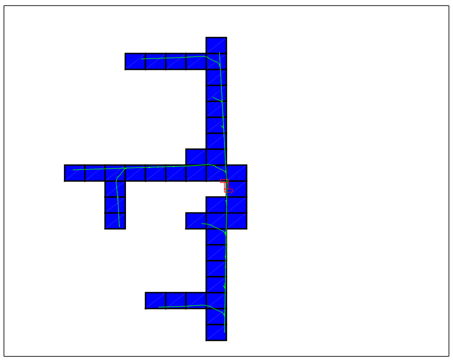
(B) S-shape_1 & S-shape_3 after removal of modules

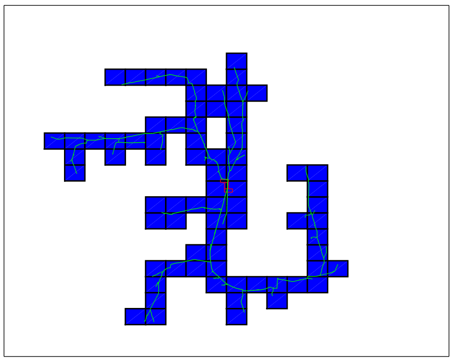
(C) S-shape_1 (fix-joint type of failure) & S-shape_3 (free-joint type of failure)

(D) S-shape_2 & S-shape_4 after removal of modules

FIGURE 4.9: S-shape construction of CoSMO robot.


(A) Percentage of visited are is 13.50%.


(B) Percentage of visited are is 10.50%.


(C) Percentage of visited are is 19.00%.

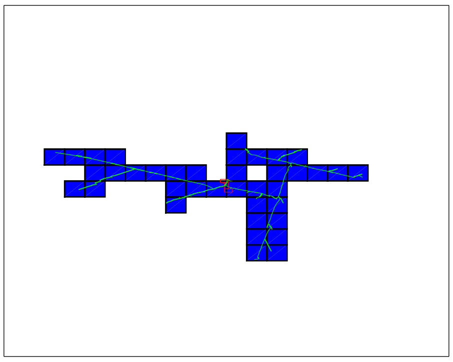

(D) Percentage of visited are is 9.75%.

FIGURE 4.10: The RRT motion planning used at S-shape_1 (on the top-left), S-shape_2 (on the top-right), S-shape_3 (on the bottom-left) and S-shape_4 (on the bottom-right) robots.

Chapter 5

Conclusion

Modular robotics is field of study of modular based robots. The modules are building components from which a robot is constructed. The modular robots are robots with many DOFs which allow them to easy step over barrier and undergo uneven ground. Thus, this technology is very useful and it could be used in missions which require adaptability and flexibility, e.g. space missions and the missions which are too dangerous for humans. The modular robots seem to be a good choice, because they can cope with and adapt on uneven and unknown environments. This characteristic is based on their ability to reconfigure / self-reconfigure.

In modular robotics there can be detected two kinds of failures: electrical and mechanical. The example of electrical failure can be that module cannot communicate with other modules or the CPU is unable to compute. The mechanical problem could be that the hinge of module is stuck or the motor does not work. Another example of mechanical failure may be that the module cannot dock with other modules.

For the purpose of this thesis the CPG-based locomotion was optimized using PSO. PSO algorithm is an optimization algorithm based on the inspiration from organisms lived in swarms. Firstly, the basic PSO algorithm was implemented on the benchmark function. Then the PSO algorithm was upgraded in order to meet the criteria of optimization of CoSMO robots. The fitness function was formulated and implemented. For purpose of the motion planning, the RRT method was implemented. First for mobile robots and later for modular robots. The RRT was modified so it utilizes the CPG locomotion to create the motion plans.

Moreover, the simulator of modular robots was created. The ODE library was used for simulation. The final step of implementation was the adaptation of the algorithm that works with robots under failures.

The goal of this thesis was to investigate influence of failures to locomotion of modular robots. The behavior of three CoSMO constructions was investigated by many experiments. It was the Cross, Dog and S-shape construction of CoSMO robot. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated. MetaCentrum was used for calculating of experiments because the computations take a lot of time and the MetaCentrum allows to use many CPUs in same time.

Most of the results of experiments show that it is better to let the robots operate with broken modules as without them. This conclusion is little surprising but on the other hand it makes sense because if the robot had smaller body his ability to move in primitive would decrease or could absolutely disappear.

Bibliography

[1] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: self-reconfigurable modular robotic system," *IEEE/ASME Transactions on Mechatronics IEEE/ASME Trans. Mechatron.*, vol. 7, no. 4, p. 431–441, 2002.

[2] P. Moubarak and P. Ben-Tzvi, "Modular and reconfigurable mobile robotics," *Robotics and Autonomous Systems*, vol. 60, no. 12, p. 1648–1663, 2012.

[3] H. Kawano, "Complete reconfiguration algorithm for sliding cube-shaped modular robots with only sliding motion primitive," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[4] C. Unsal, H. Kiliccote, and P. K. Khosla, "A modular self-reconfigurable bipartite robotic system: implementation and motion planing," *Sensor Fusion and Decentralized Control in Robotic Systems II*, 1999.

[5] "Laboratory of autonomous robotics and artificial life."

[6] "Arbeitsbereich technische aspekte multimodaler systeme."

[7] "Shape-shifting mobile robot."

[8] R. Gro, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous self-assembly in swarm-bots," *IEEE Trans. Robot. IEEE Transactions on Robotics*, vol. 22, no. 6, p. 1115–1130, 2006.

[9] A. Kawakami, A. Torii, K. Motomura, and S. Hirose, "Smc rover: planetary rover with transformable wheels," *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, p. 157–162.

[10] Z. Guanghua, D. Zhicheng, and W. Wei, "Realization of a modular reconfigurable robot for rough terrain," *2006 International Conference on Mechatronics and Automation*, p. 2999–3004, 2006.

[11] H. Brown, J. V. Weghe, C. Bererton, and P. Khosla, "Millibot trains for enhanced mobility," *IEEE/ASME Transactions on Mechatronics IEEE/ASME Trans. Mechatron.*, vol. 7, no. 4, p. 452–461, 2002.

[12] J. Liu, Y. Wang, B. Li, S. Ma, and D. Tan, "Center-configuration selection technique for the reconfigurable modular robot," *Science in China Series F: Information Sciences SCI CHINA SER F*, vol. 50, no. 5, p. 697–710, 2007.

[13] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robot. Automat. Mag. IEEE Robotics Automation Magazine*, vol. 14, p. 43–52, Mar 2007.

[14] "Robotron: Market analysis of ckbots," Oct 2008.

[15] H. Zhang, J. Gonzalez-Gomez, Z. Me, S. Cheng, and J. Zhang, "Development of a low-cost flexible modular robot gz-i," *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2008.

[16] "Ieee xplore digital library."

[17] M. Yim, Y. Zhang, K. Roufas, D. Duff, and C. Eldershaw, "Connecting and disconnecting for chain self-reconfiguration with polybot," *IEEE/ASME Transactions on Mechatronics IEEE/ASME Trans. Mechatron.*, vol. 7, no. 4, p. 442–451, 2002.

[18] Y. Li, H. Zhang, and S. Chen, "A four-legged obot based on gz-i modules," *2008 IEEE International Conference on Robotics and Biomimetics*, 2009.

[19] M. Park, S. Chitta, A. Teichman, and M. Yim, "Automatic configuration recognition methods in modular robots," *The International Journal of Robotics Research*, vol. 27, p. 403–421, Jan 2008.

[20] M. Jorgensen, E. Ostergaard, and H. Lund, "Modular atron: modules for a self-reconfigurable robot," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*.

[21] B. Salemi, M. Moll, and W.-M. Shen, "Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[22] "Crystal robot."

[23] "An interview with professor seth glodstein."

[24] "Semantic scholar."

[25] "Usd modular robotics research lab."

[26] J. Liedke, R. Matthias, L. Winkler, and H. Worn, "The collective self-reconfigurable modular organism (cosmo)," *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2013.

[27] S. Kernbach, R. Thenius, P. Corradi, L. Ricotti, E. Meister, F. Schlachter, K. Jebens, M. Szymanski, J. Liedke, D. Laneri, and et al., "Symbiotic robot organisms," *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems - PerMIS '08*, 2008.

[28] V. Vonasek, S. Neumann, D. Oertel, and H. Worn, "Online motion planning for failure recovery of modular robotic systems," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[29] Q. Wu, C. Liu, J. Zhang, and Q. Chen, "Survey of locomotion control of legged robots inspired by biological concept," *Sci. China Ser. F-Inf. Sci. Science in China Series F: Information Sciences*, vol. 52, no. 10, p. 1715–1729, 2009.

[30] Q. Liu, J. Zhao, S. Schutz, and K. Berns, "Adaptive motor patterns and reflexes for bipedal locomotion on rough terrain," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 3856–3861, SepOct 2015.

[31] D. J. Christensen, A. Spröwitz, and A. J. Ijspeert, "Distributed online learning of central pattern generators in modular robots," *From Animals to Animats 11 Lecture Notes in Computer Science*, p. 402–412, 2010.

[32] K. E. Kinnear, *Genetic programming and emergent intelligence*, p. 75–98. MIT Press, 1994.

[33] I. Tanev, T. Ray, and A. Buller, "Automated evolutionary design, robustness, and adaptation of sidewinding locomotion of a simulated snake-like robot," *IEEE Trans. Robot. IEEE Transactions on Robotics*, vol. 21, no. 4, p. 632–645, 2005.

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, p. 1942–1948.

[35] J. F. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence*. San Francisco: Morgan Kaufmann Publishers, 2001.

[36] Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, p. 81–86.

[37] S. M. LaValle, *Sampling-Based Motion Planning*, p. 185–248.

[38] S. M. Lavalle, "Motion planning," *Planning Algorithms*, p. 63–65.

[39] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat. IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, p. 566–580, 1996.

[40] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Proceedings of International Conference on Robotics and Automation*.

[41] L. Janson and M. Pavone, "Fast marching trees: A fast marching sampling-based method for optimal motion planning in many dimensions," *Springer Tracts in Advanced Robotics Robotics Research*, p. 667–684, 2016.

[42] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 2072–2078, 2015.

[43] S. M. LaValle, J. J. Kuffner, and Jr., "Rapidly-exploring random trees: Progress and prospects," 2000.

[44] T. Ju, S. Liu, J. Yang, and D. Sun, "Rapidly exploring random tree algorithm-based path planning for robot-aided optical manipulation of biological cells," *IEEE Transactions on Automation Science and Engineering IEEE Trans. Automat. Sci. Eng.*, vol. 11, no. 3, p. 649–657, 2014.

[45] W. R. Franklin, "Pnpoly - point inclusion in polygon test w. randolph franklin (wrf)."

[46] A. Yershova and S. M. LaValle, "Improving motion-planning algorithms by efficient nearest-neighbor searching," *IEEE Transactions on Robotics*, vol. 23, pp. 151–157, Feb 2007.

[47] R. Smith, "Open dynamics engine - home."

[48] V. Vonásek, K. Košnar, and L. Přeučil, "Motion planning of self-reconfigurable modular robots using rapidly exploring random trees," *Advances in Autonomous Robotics Lecture Notes in Computer Science*, p. 279–290, 2012.

Appendix A

CD content

./BT.pdf	This bachelor thesis.
./code	Contains C/C++ source files of implemented programs.