Transformations of Arm-Z modular manipulator with Particle Swarm Optimization

Machi Zawidzki* and Jacek Szklarski

Department of Intelligent Systems,
Institute of Fundamental Technological Research,
Polish Academy of Sciences, Warsaw, Poland

Dated: December 2018

Abstract

Arm-Z (AZ) is a concept of innovative hyper-redundant manipulator, which represents larger class of reconfigurable modular construction systems. AZ belongs to the family of Extremely Modular Systems, i.e. it is composed of a single basic unit and allows for creating free-form shapes.

A required level of usefulness and efficiency are among the most challenging design aspects of such reconfigurable systems to achieve. Here AZ is considered in the context of kinematics of robotic arms. Due to highly non-linear nature of the system, it may be very difficult to find transitions between two given states, especially under practical environmental and structural constraints.

This paper presents an implementation of Particle Swarm Optimization (PSO) for finding transitions between AZ states in realistic scenarios. Four practical examples are presented which are variations of two distinct problems: bending of a hexagonal AZ in a narrow slot (strong environmental constraints), and reaching a given geometrical point by the tip of a dodecagonal AZ (acting as a robotic arm). The problem of AZ transformation has been defined as a multi-objective optimization. The methodology is general with no restrictions to the objective function. Since the problem is strongly non-linear, in order to cover large space of potential solutions, the algorithm runs for a relatively large number of random initial swarms. This task was distributed on a cluster. Although the nature of AZ reconfiguration is discrete, the optimization algorithm is continuous.

Keywords: Extremely Modular System, Arm-Z, Pipe-Z, discrete optimization, dihedral rotation, reconfiguration, Particle Swarm Optimization, redundant robot, hyper-redundant manipulator.

1 Introduction

The concept of Extremely Modular System (EMS) has been introduced in 2016 [1]. The purpose of EMS is to create free-form objects with as few types of modules as possible. Chronologically, the first such system introduced in 2012 was Truss-Z (TZ for short) in [2, 3], with the purpose of creating free-form self-supporting skeletal *ramp structures* for pedestrian traffic. A year later, Pipe-Z (PZ for short) has been introduced in [4] with the purpose of creating more fundamental (abstract) objects, namely *mathematical knots*. TZ and PZ use: two and one type of modules, respectively. Arm-Z (AZ for short) is an extension of PZ introduced in [5]. AZ is a conceptual manipulator composed of congruent modules each having one degree of freedom (1-DOF) - a

*Electronic address: zawidzki@mit.edu; Corresponding author

relative twist. Simple changes of these twists result in emerging behavior of the entire AZ allowing it to perform complex movements. Figure 1 shows a physical model of two knots.

Figure 1: Two physical models of mathematical knots constructed with just one type of dodecagonal module. *Trefoil* and *Cinquefoil* are shown on the left and right, respectively. Initial module and direction are indicated by gray and arrow, respectively. For the ease of assembly the module has male and female sides, as shown in the inset in the middle.

PZ, the predecessor of AZ has been proposed as a deployable construction system e.g.: for space habitats and emergency connectors [6]. In that paper the deployment has been based on synchronous unfolding of each PZ module [see Figure 5]. However, alternatively, a change of overall shape can also bee considered as a deployment mechanism. In this paper, In order to demonstrate exceptional capabilities of AZ as a modular reconfigurable system, it has been faced with selected challenges or robotic manipulators, which are known to be the most stringent among reconfigurable systems.

The basic module in both systems (AZ & PZ) is a geometrical object analog to a sector of circular torus described in [7]. It is defined by parameters: r: (0, ∞), d: (0, ∞) and ζ: (0,∞), which denote: radius, corresponding radius and central angle, respectively; r, d, $\zeta \in \mathbb{R}$. Each module is terminated by two faces T and B, corresponding to the top and the bottom of a unit. Although in principle T and B do not have to be congruent, for practicality, however, it is desirable that the modules are symmetrical about the plane perpendicular to their axes. Such a condition implicates that T and B are congruent. Their relative position is controlled by r, d and ζ. The faces of T and B can have shapes of circles or regular polygons of arbitrary number of k sides. Moreover, it is convenient to introduce a new parameter $s = \frac{d}{r}$; $s : (0, \infty)$. Thus r is the "absolute" parameter controlling the size of the module in relation to the environment, and s is the "relative" parameter defining its "slenderness". Figure 2 illustrates these relationships.

The forms shown in Figure 1 resemble biological snakes. The snake mechanism is a redundant system which, however, makes them supremely adapted for the habitats. Analogously, in irregular environments, bio-inspired snakelike robots in some cases may perform better than the more conventional wheeled, tracked and legged forms of robotic mobility. Research on snake robots has been conducted for several decades. Snake locomotion has been studied empirically already in 1940s [8]. 50 years later, the first mathematical model has been developed and snake-like locomotors and manipulators have been proposed in [9]. Snakelike (or "trunk-like") manipulators have specific movement characteristics which makes them particularly useful in situations where classic robotic arms can not be used (e.g., in complex, narrow, constrained environments). Additionally, snakelike manipulators may have relatively large number of degrees of freedom (however, it is not always the case). This contrasts with the classic robotic arms with low number of DOFs which are widely used in the industry.

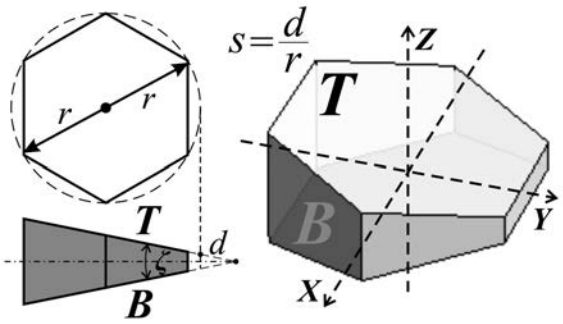

Figure 2: A visualization of basic hexagonal module which is defined by parameters: r, d, ζ and $k = 6$. The axonometric view on the right indicates the placement of the initial module in the coordinate system.

In the case of AZ manipulator, the number of DOF is obviously equal to the total number of the joined modules. Apart from the possibility of performing complex actions, large number of DOF introduces the possibility of high fault tolerance and robustness. It aligns well with the concept of, so called, hyper-redundant manipulators (HRM, [10]). Figure 3 shows some examples of HRMs. The advantages of manipulators of this type come at a price of notoriously difficult control due to the highly nonlinear nature of such systems.

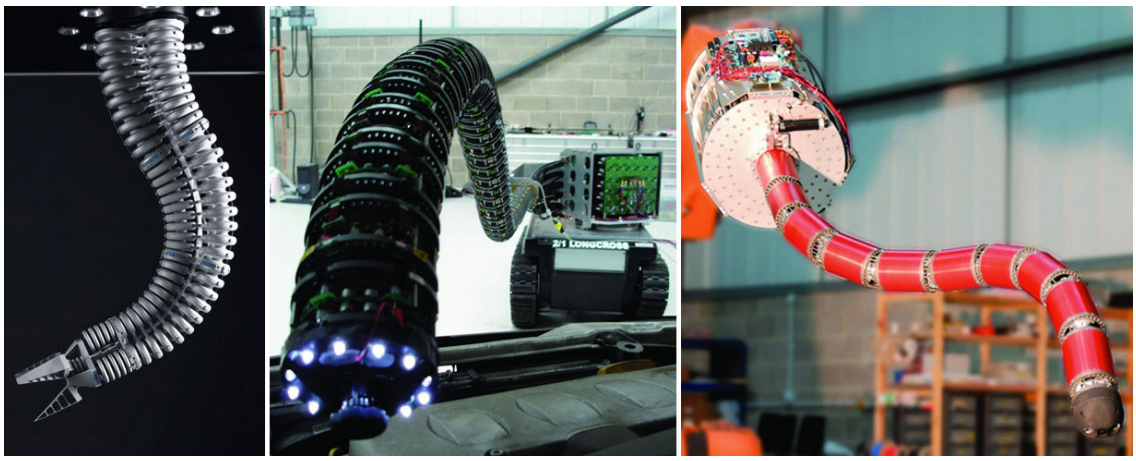

Figure 3: From the left: Festo (12 DOF; http://www.festo.com), and two "snake-arms" by OC Robotics (http://www.ocrobotics.com).

For example, the inverse kinematics problem for a typical industrial robotic arm can be solved relatively easy (e.g. [11]) and therefore such arms are easy to control. On the other hand, the control of a *bionic trunk* requires sophisticated artificial intelligence methods [12–14]. However, in return, one can profit from all the important advantages of snakelike HRMs. For further details on this interesting class of robotic manipulators see [15], regarding details on AZ and other Extremely Modular Systems see [1].

This paper is an updated and revised version of the conference paper [16], which in the nutshell, introduced the idea of discrete reconfiguration of a modular structure, formulated the problem as combinatorial, applied an exhaustive search method and implemented parallelization to speed up the computation. As an example the preliminary optimization of reconfiguration of a modular manipulator from a "straight tube" to a half-torus was presented. In this problem, the "wobbling" of the manipulator to the sides was subjected to minimization. In this case the relative twists of the modules were discrete, thus the approach was combinatorial in nature. Parallelized brute-force

search method has been applied for finding the ideal solution for a small manipulator comprised of eight hexagonal modules. The efficiency of parallelization was shown to be very good. The new contributions of the current paper can be summarized as follows:

- New representation of transition between the states of AZ by piecewise velocities. Such representation allows for application of well established heuristic methods.
- Particle Swarm Optimization (PSO) is applied for Arm-Z transitions (AZT, for short).
- The AZTs are formulated as constrained multi-objective optimization.
- The new formulation of the problem allows for efficient and well-scalable parallelization [17], which is demonstrated with practical examples.
- Four cases are presented:
  1. finding the same ideal solution with PSO as the one presented in [16], and therefore proving that the PSO can converge to the known global optimum in this case;
  2. minimization of wobbling of a 16-unit manipulator which bends from a "straight tube" to a full torus which is a representation of a complex transition in a tight, constrained environment. The result is compared with previous solution obtained by means of straightforward greedy methods;
  3. transition from a given starting configuration ("straight tube") to a target configuration in a constrained environment (the final configuration is given), intersection with obstacles is taken into account;
  4. as 3, however, the target configuration is not given explicitly, instead the center point of the last AZ unit is expected to get as close as possible to a target point. This is the most complex case which in principle can be used to control the head of AZ in any kind of environment.

2 The AZT problem

The AZ Modules are based on polygons, which results in non-continuous ("step") motion. For discussion on dihedral systems, especially based on regular (Platonic) tessellations see [18]. Such discrete motion is rather unnatural, since manipulators are usually designed to cover continuous space within given range and to perform motion as "smoothly" as possible. Nevertheless, it is an early stage of AZ development and it has been decided for its discrete nature for the following reasons:

- The AZ configurations can be described as a sequence of dihedral rotations and expressed by a list of integers.
- It is relatively easy to fabricate an inexpensive mock-up model of AZ which can firmly hold shape in any configuration without any energy supply (see Figure 4).
- Step-motors are considered to power the relative twists of AZ. They also support step-motion, which, however, has much higher resolution.
- It is much easier and less expensive to fabricate and fold a module flat ([6]) which has rather faceted surface than curved, as shown in Figure 5.

Obviously, for an AZ based on a polygon of n sides (n-gon), the number of possible relative twists for each module is equal to k. Figure 6 shows two examples: dodecagonal and hexagonal. Thus a "straight pipe" for: square, hexagonal, octagonal, decagonal etc. AZ will be encoded as a sequence of integers: $[2, 2, \dots]$, $[3, 3, \dots]$, $[4, 4, \dots]$, $[5, 5, \dots]$, respectively. On the other hand, for

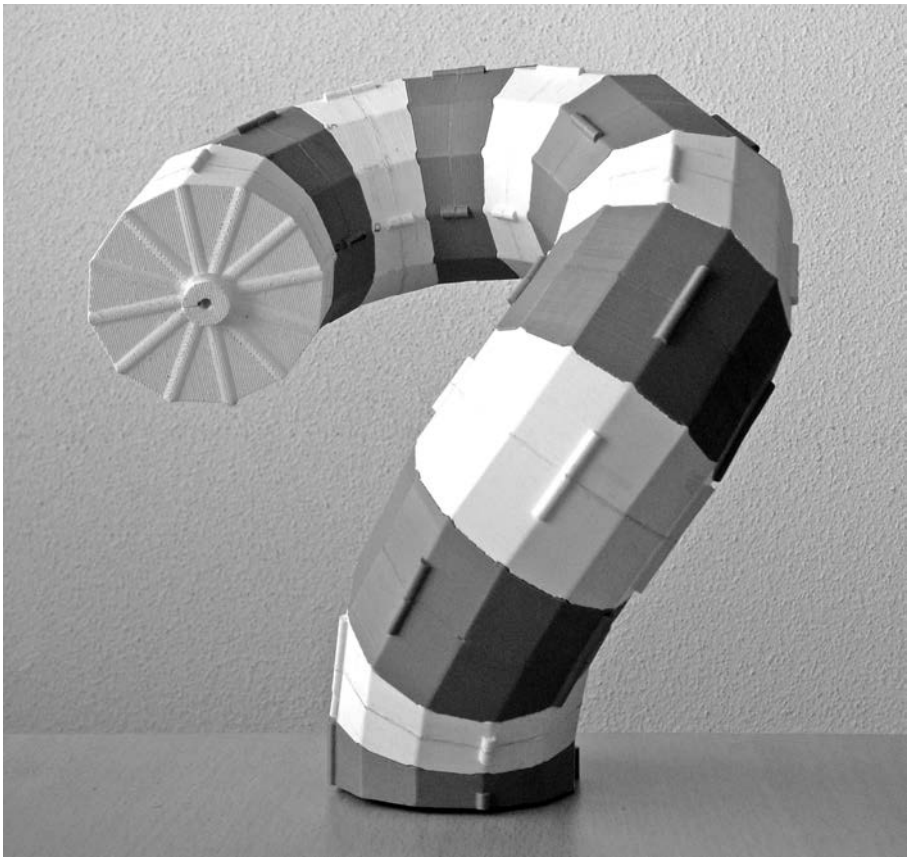

Figure 4: A photograph of a 3D-printed mock-up prototype of a dodecagonal 12-module AZ.

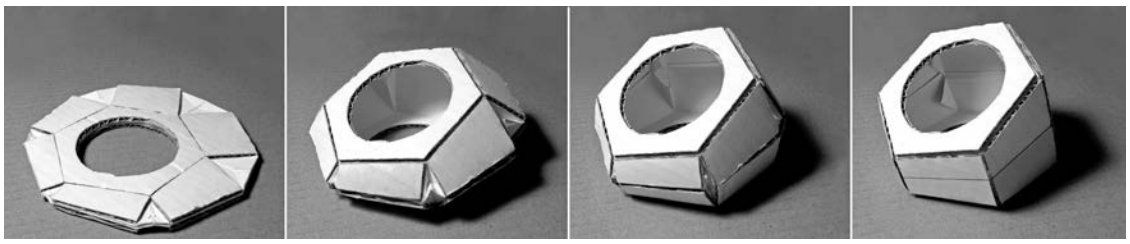

Figure 5: Four stages of origami-inspired unfolding of a physical low-tech paperboard model of a hexagonal module.

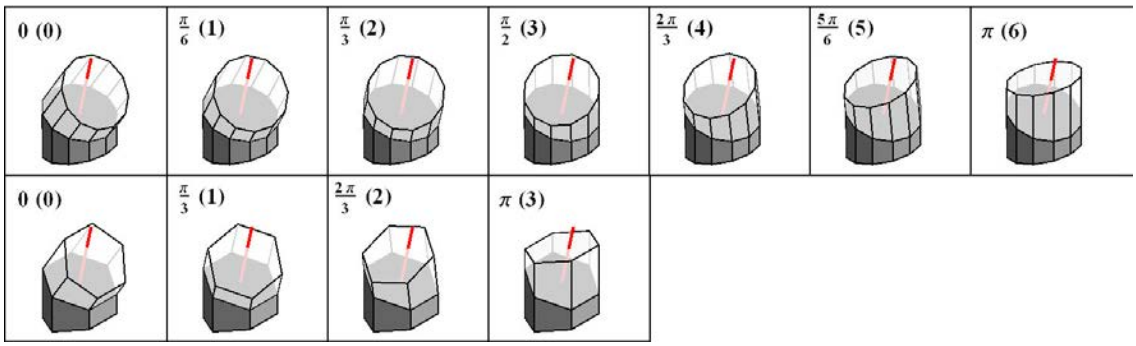

Figure 6: Two AZ modules of different base at consecutive dihedral twists from 0 to π. On the top and bottom: dodecagonal and hexagonal AZs are shown, respectively. The numbers in parentheses indicate the consecutive positions. For simplicity, further in text, this notation will be used. The first (bottom) module is fixed and indicated in gray. The axis of rotation for the second module (white partially transparent) is shown in red.

an AZ of any base, a sequence of 0's will form an arc up to a full torus (compare with Figures 7 and 14).

In principle, the problem of AZT is defined as follows: provide the sequence of relative twists for each module in AZ which transform given initial configuration $\mathbf{S_s}$ to another configuration $\mathbf{S_e}$. Such a transition can be subjected to optimization (e.g., minimization of the number of steps) and/or constraints (e.g., by obstacles in given environment). The configurations $\mathbf{S_s}$ and $\mathbf{S_e}$ can be given explicitly (as it is done in cases 1-3, Sec. 4) or by a set of requirements (e.g. $\mathbf{S_e}$ must be so that the last module reaches certain area and entire AZ does not violate given obstacles; case 4).

The challenges of AZT were illustrated with the case presented in the original paper [16]. The problem was defined as follows:

- There are two given configurations (states) of AZ: Start ($\mathbf{S_s}$) and End ($\mathbf{S_e}$).
- The objective was to find the AZT between $\mathbf{S_s}$ and $\mathbf{S_e}$.
- The number of transformation steps was given a priori, namely: 4.
- In one step, each module of a n-gon base AZ can rotate by at most $\frac{2\pi}{n}$ dihedral angle. E.g. each module of a hexagonal AZ at one time-step can rotate by at most $\frac{\pi}{3}$, that is by: $\{\frac{\pi}{3}, -\frac{\pi}{3}, 0\}$.
- The summarized wobbling perpendicular to the "bending plane", i.e. in the y-direction to be minimal.

Figure 7 illustrates this problem and its ideal solution. The following geometrical parameters of the basic module were used: $n = 6$, $r = 0.25$, $\zeta = \frac{\pi}{8}$ and $s = 0.4$. The center of the first element is placed in the center of Cartesian coordinate system (as shown in Figure 2).

The original approach was based on exhaustive search, which can only work for very small problems. That is because the growth of the number of all possible AZT sequences from one configuration to another is *double exponential*. For this particular problem there are two symmetrical ideal solutions among 262,144 potential solutions. One kernel of a regular PC required already approximately four hours for this computation. Although finding an ideal solution even for such simplistic case is beneficial for fine-tuning more sophisticated algorithms, this also shows that this approach is not suitable for any problems with realistic size.

The main difficulty of AZT is that this problem is highly non-linear and difficult to trace. This means that small changes in a small number of segments can lead to completely different overall shapes and for these reasons, e.g., any gradient-based methods would inevitably fail. It

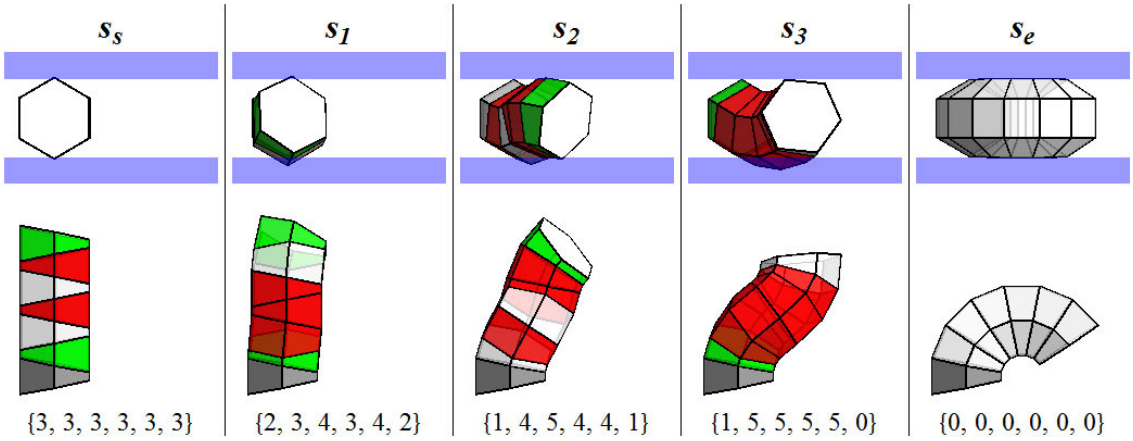

Figure 7: The ideal 4-step transition of a 7-unit hexagonal AZ from Start ($\mathbf{S_s}$) to End ($\mathbf{S_e}$) states, shown in extreme left and right, respectively. For each state the top and side views and the sequence of positions of the modules are shown. The first unit in the sequence is fixed and is shown in gray, it also is omitted in the list of twists. Green and red indicate the twists in the next time-step to: the right and left, respectively. Blue stripes restrict the allowable slot.

is therefore rational to apply a heuristic method which can quickly sample as large part of the parameter space as possible.

Heuristic methods, in particular based on evolutionary algorithms have been successfully applied to EMS optimization. E.g. the layouts of single- and multi-branch TZ structures have been optimized with genetic algorithms in [19] and [20], respectively. Graph-theoretic approach has also been applied for finding *ideal* solutions of a single-branch TZ layout in a highly constrained environment as presented in [21]. In this paper, however, another classic heuristic method is used for AZT optimization, namely Particle Swarm Optimization (PSO).

Since its introduction in 1995, PSO [22] has been successfully used in various areas, e.g.: transportation [23], engineering [24], medicine [25], planning [26] and many others [27–29]. Moreover, PSO has been proved to be efficient at solving highly nonlinear control problems [30, 31] and therefore has been chosen for AZT. Using PSO for each transition problem, a relatively large number of initial conditions (swarms) can be considered, and since the computing threads are independent, the calculations can be easily parallelized, including massive parallelization with GPU [32, 33]. As a result, as showed the experience, for realistic problems acceptable solutions can be found in a matter of minutes/hours, depending on their complexity. Note that by appropriate optimization of the implementation, there is a significant potential for improvement and decrease the computational time substantially making the control possible in real-time.

3 Particle Swarm Optimization

PSO is a meta-heuristic optimization method which iteratively tries to improve a candidate solution. As typical methods of this sort, it does not guarantee that the optimum is found, however, a very large space can be searched and there are no requirements regarding the objective function. Therefore PSO can be used for irregular, noisy, coarse problems, or multi-objective optimization.

3.1 The encoding of AZT

Before applying PSO, it is necessary to define an AZ transition in a form suitable for use with this optimization method. Transition is the process of changing an initial state $\mathbf{S_s}$ into the final state $\mathbf{S_e}$. $\mathbf{S_e}$ can be given explicitly as a vector describing states of each module, or as a criterion which the final AZ must meet (e.g., position the last module as close as possible to a given point

in space). AZT may be subjected to additional constraints, i.e., prohibition of self-intersections or collisions with existing obstacles in the environment.

Let n be the number of sides of the AZ module base, and l be the total number of modules. Without loosing generality, let us consider a single rotating module. Its state can be represented by an integer s, $s \in [0, \ldots, n-1], s \in \mathbb{I}$. However, for the purpose of encoding solution for reconfiguration, a corresponding continuous counterpart $\tilde{s}$ is introduced, $\tilde{s} \in \mathbb{R}$. The discrete value representing the real integer state is the nearest-integer, $s = \lfloor \tilde{s} \rceil$ (here $\lfloor \rceil$ denote the nearest-integer operator, e.g., $\lfloor 4.6 \rceil = 5, \lfloor 2.5 \rceil = 2$)

In order to allow for a change of the shape of AZ, each element i is assigned an angular velocity $\omega_i(t)$ which maybe time-dependent (thus acceleration and deceleration are possible). The time runs from t_s to t_e where t_s and t_e correspond to the initial and end states, respectively. Therefore the discrete states of the $i-$th element during the transition are given as:

$$s_i(t) = \lfloor \tilde{s}(t) \rceil = \left\lfloor \int_{t_0}^{t} \omega_i(t) \mathrm{dt} + s_i^s \right\rceil$$

where s_i^s is the initial state at time t_s.

Consequently, the total transition is controlled by the velocity functions $\omega_i(t)$. It is assumed that the transition time is divided into k equal parts ("number of intervals"), $\delta t = (t_e - t_s)/k$ during which the velocity is kept constant, k being a parameter. Therefore the resulting real-valued function $s_i(t)$ is piecewise linear, and ω_i is simply defined by a real valued vector with k elements. For the given k (which is the same for all the segments), the transition for the entire system is then defined by a matrix $\mathbf{\Omega}$ of $k \times l$ real values for the given time span $t_e - t_s$. Note that the assumption that the k parts are equal is arbitrary. In principle the transition time could be divined into parts of any length. However, this would introduce additional variables into the optimization process making it more difficult due to the "curse of dimensionality".

As a result, for $k = 1$ each elements will linearly changes its state in one direction (depending of course on the sign of ω). Increasing k to 2, will make it possible that, e.g., rotation will stop after δt, it also can change direction, allowing for more complex transition. The drawback is that in this case, the search space for ω will be increased by a factor of 2. Figure 8 shows an example of $s(t)$, $\tilde{s}(t)$ and $\omega(t)$ for a single segment transition, $t_s = 0, t_e = 3$ with $k = 3$, resulting in the sequence [3,4,3,2].

The purpose of the optimization is to find the values of the matrix $\mathbf{\Omega}$. The discrete solution will be a matrix of the form $\mathbf{S} = [\mathbf{S_s}, \mathbf{S_1}, \mathbf{S_2}, ..., \mathbf{S_e}]$, where $\mathbf{S_j}$ are vectors of integers denoting the discrete states of all the AZ unit. It is assumed that consecutive states $\mathbf{S_j}$ and $\mathbf{S_{j+1}}$ will differ by at least one element and each element's state will differ by at most 1. These assumptions together with $\mathbf{\Omega}$ and given t_s and t_e will determine the total discrete transition $\mathbf{S}$.

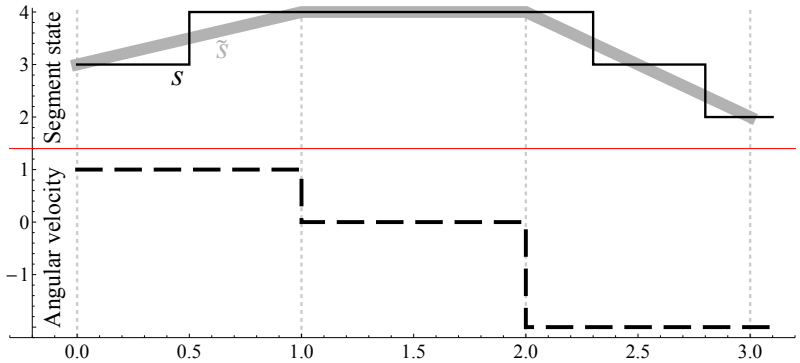

Figure 8: A sample plot showing $s(t), \tilde{s}(t)$, and $\omega(t)$ for $k = 3$, $\omega_i = [1, 0, -2]$ and $s_i^s = 3$.

3.2 Implementation of PSO for ATZ

In the presented approach, the classical version of PSO is used to find the values of the matrix $\mathbf{\Omega}$. The process of relating transition to a particle in PSO is referred to as "coding" (Subsec. 3.1). Since the transition is entirely defined by the matrix $\mathbf{\Omega}$, the particle will simply correspond to the consecutive elements of $\mathbf{\Omega}$. The limit on the velocities is imposed by requiring that each component of the vector to be in range $-1, \leq x_j \leq 1$. This sets a scale for the problem: for example, a module with $k = 6$ sides with $\omega = -1$, $t_s = 0$, $t_e = 2.1$, and the initial state $s = 0$ will end up in the state $s_e = 4$.

With each particle $\mathbf{x_i}$, there is associated a vector called its velocity $\mathbf{v_i}$ (do not confuse with the transition angular velocities), whose elements lie in the range $[-2, 2]$. Additionally, each particle i keeps track of its best position in the history of optimization $\mathbf{p_i}$, that is a position for which $f(\mathbf{p_i})$ is extreme (minimal or maximal, depending on the desired optimization objective).

Let $\mathcal{N}$ be the set (the swarm) of N particles representing N solutions ($\mathbf{\Omega}$'s). The PSO algorithm iteratively moves all the particles $\mathbf{x_i}$ through the search space according to theirs velocities $\mathbf{v_i}$. Each particle is attracted to its all-time best position $\mathbf{p_i}$ and to the swarm all-time best solution $\mathbf{b}$ (updated at each step). The degree of this attractiveness, along with the particle "intention" to follow its velocity, are defined by the parameters ϕ_p, ϕ_b and ω_g respectively. The PSO algorithm for finding optimal, in the sense of minimizing some utility function $f(\mathbf{x})$ is presented in Tab. 1.

The definition of the minimized objective function $f(\mathbf{x})$ obviously depends on the desired optimization objective. Typical variables used to define the goal may include:

- D – is the Cartesian norm between the final state encoded by a particle $\mathbf{S_x}$ and the given $\mathbf{S_e}$ if the desired end-state $\mathbf{S_e}$ is given beforehand,
- d – is simply the geometric distance between the tip-point $\mathbf{p_e}$ of the last segment and the given target point $\mathbf{p_t}$,
- w_i – is the total maximal distance between a centroid of the $i-$th segment for two consecutive transitions, (i.e., two columns of $\mathbf{S}$). Minimizing w will prevent undesired, potentially mechanically harmful jumps between states,
- o – denotes number of centroids being *outside* an operational volume during the transition. This prevents the AZ from hitting obstacles or crossing itself. In principle, the operational volume may change during AZT, however, here only static cases are considered.

One of the issues of fundamental importance when using PSO is the proper selection of the control parameters ω_g, ϕ_p, ϕ_b and the population size N. Since PSO is intrinsically flexible and indeterministic, it is impossible to give a formal prescription for values of these parameters which would give the best results in a general case. Although, there exist some research concerning procedures how to adjust these values [34], possibly during the optimization process [35], often they are selected by means of a trial-and-error method. In such procedure, $\omega_g, \phi_p, \phi_b, N$ are chosen arbitrary in a way which reflects the nature of the underlying objective function. One should keep in mind that, generally speaking, ω_g controls the tendency to explore the entire search space, ϕ_p the tendency to explore in the vicinity of local extrema, and ϕ_b the convergence rate to the best solution found so far. N is responsible for the diversity of potential solutions (which is also associated with the demand for computational resources).

It is clear that the PSO control parameters can be very different for various problems and it may be difficult to find satisfactory balance between exploration and convergence. In order to select values for our model, convergence rates and the ability to find the globally optimal solution for various, random initial conditions has been explored by means of a trial and error method. Eventually, the following values were used to calculate all the presented results: $\omega_g = 1, \phi_p = 1.0, \phi_b = 2.0$ and $N = 200$. Moreover, for each considered case PSO has been run for 5×10^3 various initial swarms, each evaluated for 200 generations, in order to further explore the search space.

1	For each particle i:
2	initialize $\mathbf{x_i}$ with random values, $\mu_{\min} \leq x_i, j \leq \mu_{\max}$
3	let the particle's best position $\mathbf{p_i}$ will be equal to its initial one: $\mathbf{p_i} \leftarrow \mathbf{x_i}$
4	initialize $\mathbf{v_i}$ with random values, $\eta_{\min} \leq x_i, j \leq \eta_{\max}$
5	calculate the objective function for particle i: $f_i(\mathbf{p_i})$
6	update the swarm best solution: if $f_i \leq f(\mathbf{b})$ then $\mathbf{b} \leftarrow \mathbf{x_i}$
7	For each particle i:
8	pick random numbers r_p, r_b from the range $[0, \ldots, 1]$
9	for each component j:
10	$v_{i,j} \leftarrow \omega_g v_{i,j} + \phi_p r_p(p_{i,j} - x_{i,j}) + \phi_b r_b(b_j - x_{i,j})$
11	$\mathbf{x_i} \leftarrow \mathbf{x_i} + \mathbf{v_i}$
11	if $f(\mathbf{x_i}) < f(\mathbf{p_i})$ then the particle's best $\mathbf{p_i} \leftarrow \mathbf{x_i}$
12	update the swarm best solution: if $f_i \leq f(\mathbf{b})$ then $\mathbf{b} \leftarrow \mathbf{x_i}$
13	If a given termination criterion is not met, go to 7

Table 1: The Particle Swarm Optimization algorithm. Each particle represents the transition velocities $\mathbf{\Omega}$, i.e., $l \times k$ real numbers in a given range $[-1, 1]$. Note that these transition angular velocities should not be confused with the particle's velocities in the PSO algorithm. $\mu_{\min} = -1, \mu_{\max} = 1$, $\eta_{\min} = -2, \eta_{\max} = 2$. Uniform distribution is used for drawing the random numbers.

4 Four examples of PSO applied to AZT

This section presents four examples of PSO applied to AZT, from a relatively simple case of bending of a short AZ in a narrow slot where both start and end configurations are given, to a realistic task of AZ tip reaching a given point in an environment where final AZ configuration is unknown, elements of the environment must not be violated and the wobbling of AZT to be minimal.

4.1 7-unit AZ bending in a slot in 4 steps

The conditions in this example are the same as described in Section 2. The brute-force search produced two symmetrical and equivalent ideal solutions within a few hours. The challenge here is to reproduce this result much more quickly. The following initial conditions are assumed: $t_s = 0, t_e = 5, k = 4$. The velocities are rounded to ω' as follows:

$$\omega' = \begin{cases} -1 & \text{if } \omega < -0.5 \\ 0 & \text{if } -0.5 \leq \omega \leq 0.5 \\ +1 & \text{if } \omega > 0.5 \end{cases}$$

The initial conditions along with the above rounding has been chosen in order to verify if the original 4-step solution found by means of brute-force can be reproduced. The search here is simply equivalent to minimization of the sum of the buckling error, so the objective function to be minimized is $f = \sum_{i=1,\ldots,6} |\delta_y(i) + D$, where $\delta_y(i)$ denotes the absolute difference between the y-coordinate of the centroid of the initial module and the $i-$th module (since the center of the first module lies at the coordinate center, δ_y measures deviation from the y-plane). It has been shown by means that for the parameters stated above, ideally $f_{\text{opt}} = 0.50308$, see [16], and D is of course 0 (D is used to "guide" the swarm towards the desired final state).

The optimal matrix found by PSO is:

$$\mathbf{\Omega}' = \begin{bmatrix} 1 & 0 & -1 & 0 & -1 & 1 \\ 1 & -1 & -1 & -1 & 0 & 1 \\ 0 & -1 & 0 & -1 & -1 & 0 \\ 1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

It gives the following transition: $[[3,3,3,3,3,3], [4,3,2,3,2,4], [5,2,1,2,2,5], [5,1,1,1,1,5], [0,0,0,0,0,0]]$, and was found in about 5% of random initial swarms, where each swarm was evaluated for 200 generations with the PSO parameters stated as above. An example of convergence to the optimal value is depicted in Figure 9. Figure 10 shows the angular discrete velocities of each module.

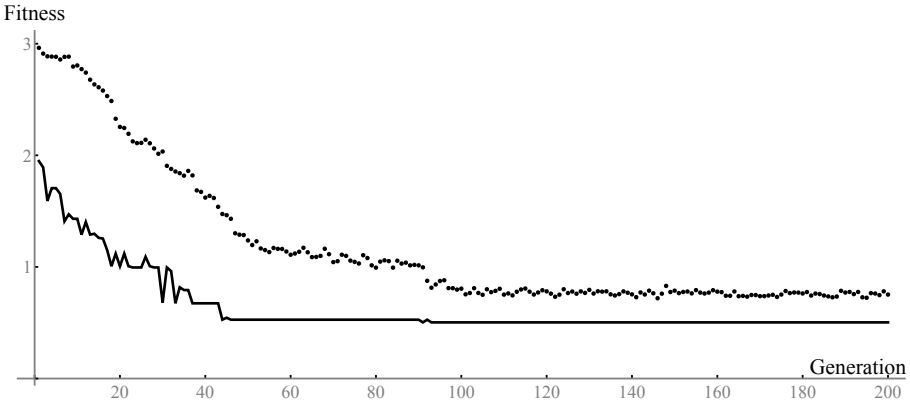

Figure 9: PSO of 7-unit AZ bending in a slot. The black dots and line indicate: swarm average and the best fitness function values, respectively. The optimum $f_{\text{opt}} = 0.50308$ is reached after ≈ 50 steps.

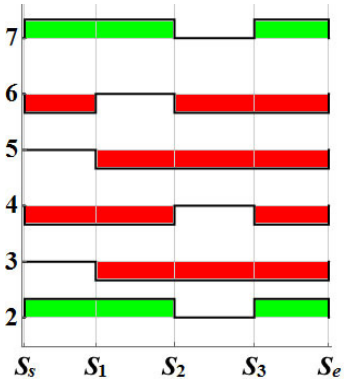

Figure 10: The angular velocities of each module of the 7-unit AZ in transition from $\mathbf{S_s}$ to $\mathbf{S_e}$. The horizontal and vertical axes correspond to: the subsequent states (configurations) and indices of the modules, respectively. Green and red indicate the relative twists in: right and left, respectively. The first module is fixed, and thus is omitted.

4.2 16-unit bending in a slot

This is an extension of the previous example. Here AZ is comprised of 9 units more, i.e. 16 in total. Considering also only 4 time-steps, there are 35,184,372,088,832 possible AZTs. This shows the complexity of the problem as well as the futility of the brute-force methods for this case. Moreover, there is no reason to the number of time-steps set beforehand, which makes the space of potential solutions in combinatorial approach simply unimaginable.

The conditions for this multi-objective PSO have been defined as follows:

- Minimize $\max \delta_y$ (only in y-plane) and D so the objective function to be minimized is $f = 5 \max \delta_y + D$, where D is the distance to the final state (of course D must be 0 for a correct solution finishing with $\mathbf{S_e}$.

- The number of steps is not pre-set and it is one of the results of the process.

The factor of 5 used above has been set arbitrarily in order to give more importance of the y wobbling during the swarm evolution described in Sec. 4.2.2. The minimization of δ_y is done for all the segments and for all the transition steps (this follows from the assumption that the maximal deviation from y should be minimal at *any* time during the transition).

4.2.1 The "greedy" approach

The first solution for this problem has been presented in [5]. It is based on a quite intuitive strategy which formally is a greedy algorithm:

- There is a parametrized guide-line (GL) of length equal to the length of a 16-unit AZ (L_{16}, for short).
- L_{16} is constant, however, the curvature of GL can change from $\mathbf{S_s}$, that is a straight line segment (i.e. an arc with an infinite radius) to $\mathbf{S_e}$, that is a full circle of radius $\frac{L_{16}}{2\pi}$. For the corresponding illustration with an interactive demonstration see [36].
- AZ is continuously aligned to GL, and the curvature of GL is gradually changed from $\mathbf{S_s}$ to $\mathbf{S_e}$. The alignment is done from the bottom to the top of GL, each unit being added with such a state which minimizes δ_y (hence "greedy"). For the details of this alignment procedure see [4].
- The AZ configurations are collected at given time-steps and the unique sequences are recorded.

Figure 11 illustrates the idea of controlling the shape of AZ by the curvature of GL.

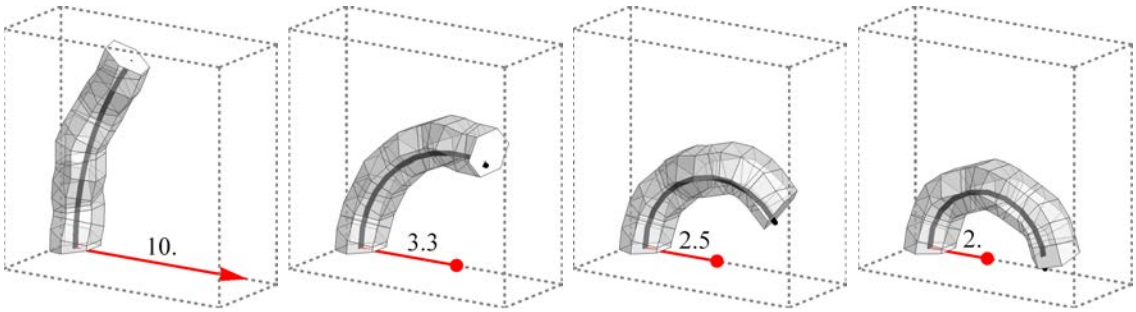

Figure 11: Four selected states of the continuous alignment of AZ to the given GL (shown as a thick black line). The radii of arcs are shown for each case and the corresponding centers are indicated in red. Here for simplicity $L_{16} = 2\pi$.

Figure 12 shows the AZT based on this intuitive approach. Figure 13 shows further analysis of this AZT. As they both indicate, the strategy of collecting local optima step-by-step is rather successful. AZ bends relatively smoothly only slightly violating the slot restriction. It is interesting, however, whether such transformation can be done at fewer steps and/or with smaller error. The following subsection describes the implementation of PSO to this problem.

4.2.2 The solution by PSO

Three different values of k were considered: 1,2, and 3; the corresponding best values of δ_y are: 0.172, 0.288, and 0.480. This is somehow surprising since, clearly, the possibility of non-zero acceleration (i.e., $k > 1$) not only does not improve the solution but it makes it significantly worse. It is attributed to the increase of number of searched variables and the associated increased difficulty of finding good solution. This would also explain why only small amount of swarms evolve

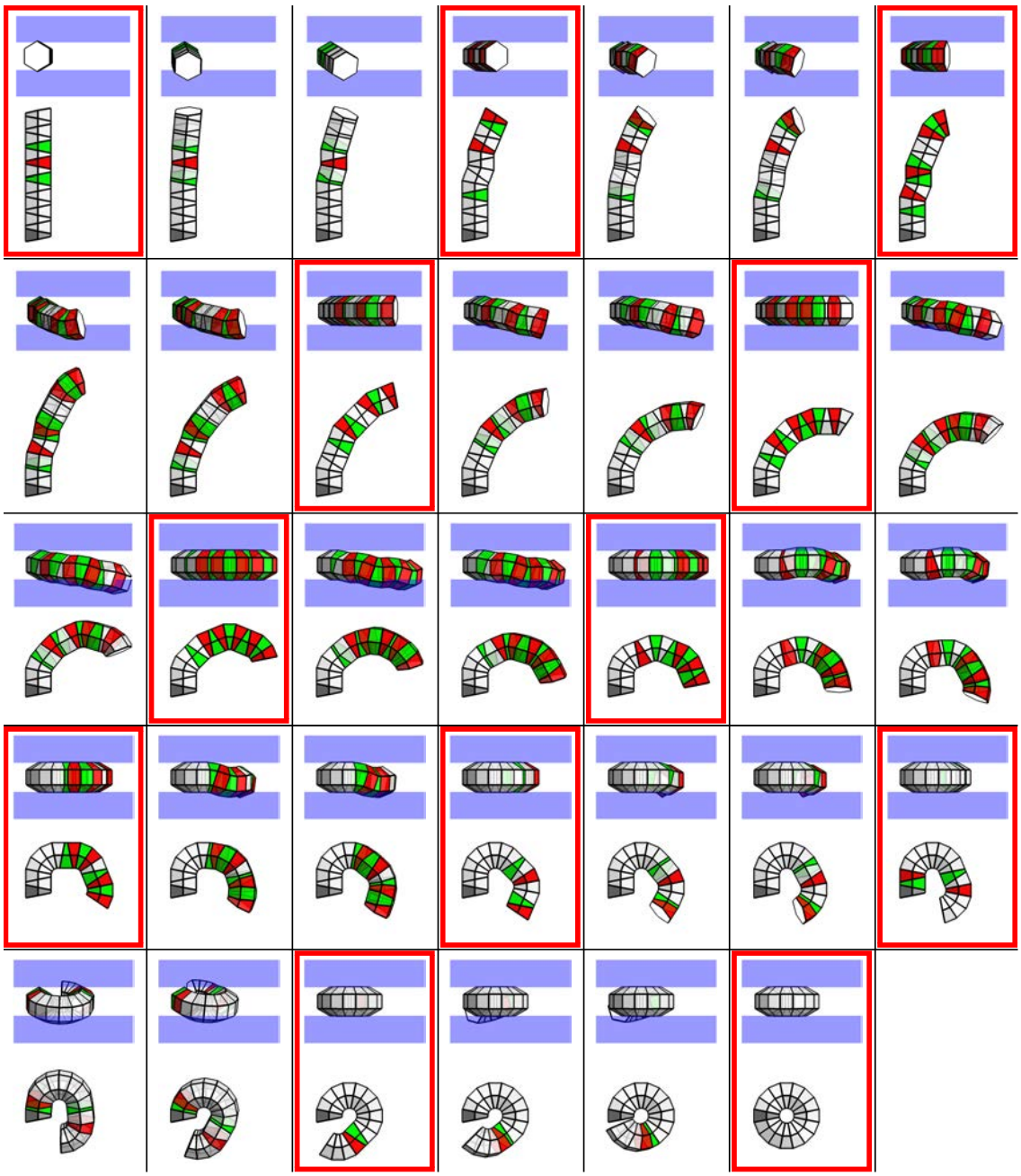

Figure 12: The AZT found by the "greedy" approach requires 33 time-steps. The color convention as in Figure 7. Red frames indicate the states where the AZT fits perfectly the given slot, in other words the error is 0. This situation is cyclical and occurs at every third time-step. That is because at this period all AZ modules are either in position 0 or 3.

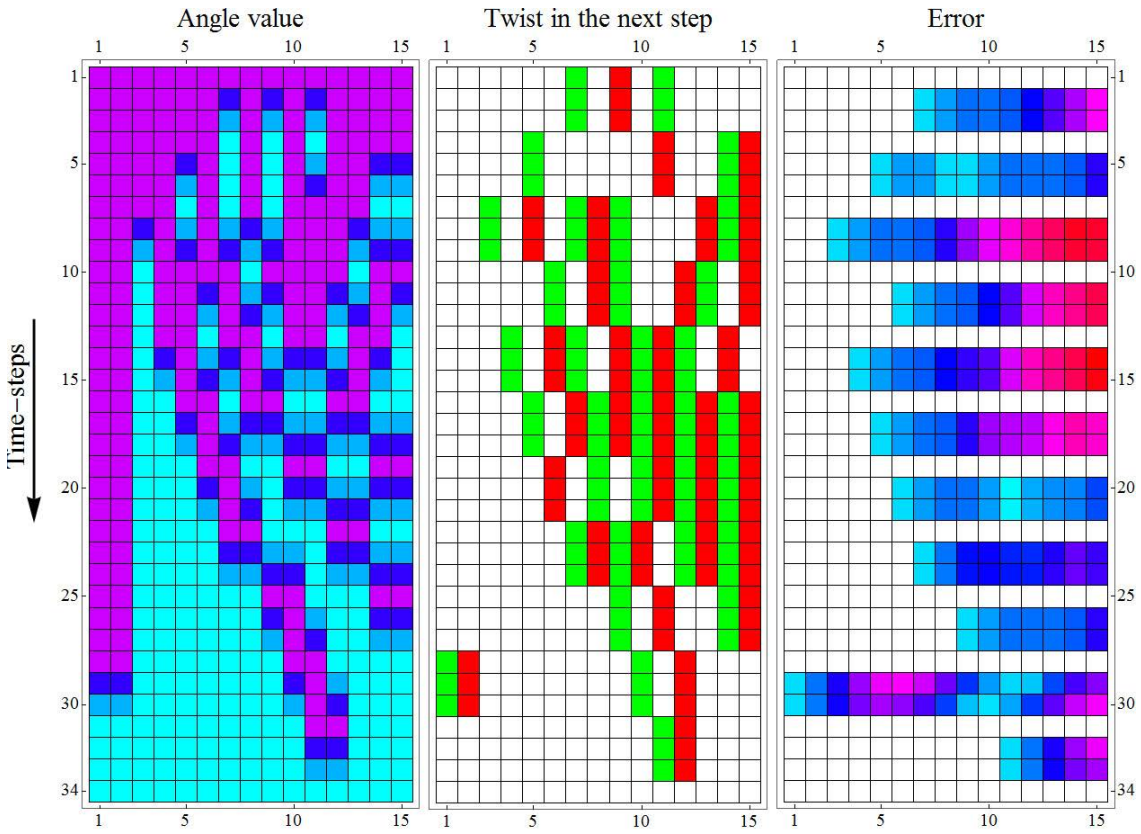

Figure 13: The values of angular parameters and errors for each module in AZ during entire "greedy" transition. The history runs from the top, i.e. the top and bottom rows represent: $\mathbf{S_s}$ and $\mathbf{S_e}$, respectively. On the left: angle values expressed as dihedral positions; magenta, navy, blue and cyan indicate positions: 3, 2, 1, and 0, respectively. In the middle: the relative twists in the next time-step; Green and red indicate the twists to the right and left, respectively. On the right: the errors; white indicates perfect alignment, whereas red indicates the largest error $\max \delta_y = 0.254$.

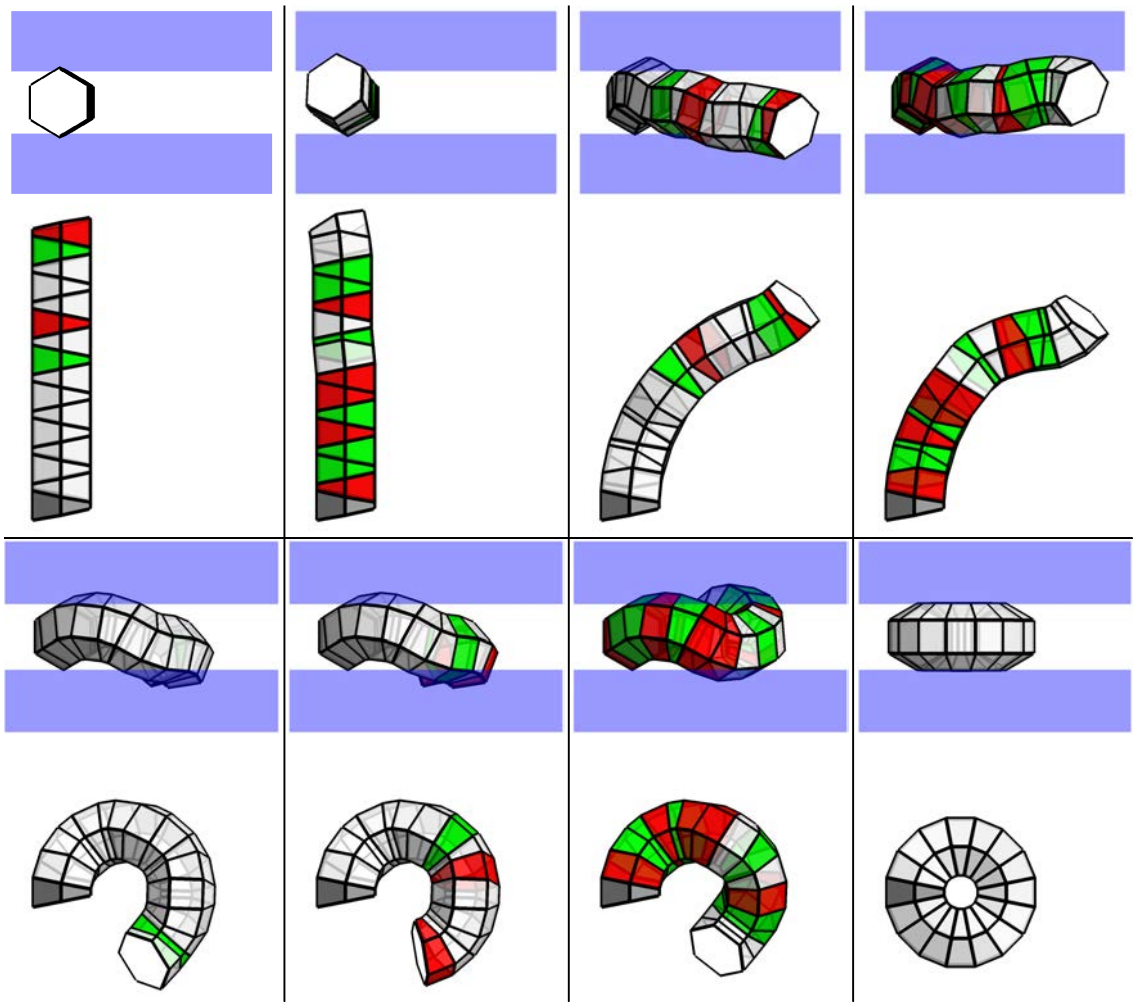

Figure 14: The best result found for the 16-unit hexagonal AZ from "straight pipe" ($\mathbf{S_s}$) to the full torus ($\mathbf{S_e}$). The color convention as in Figure 13.

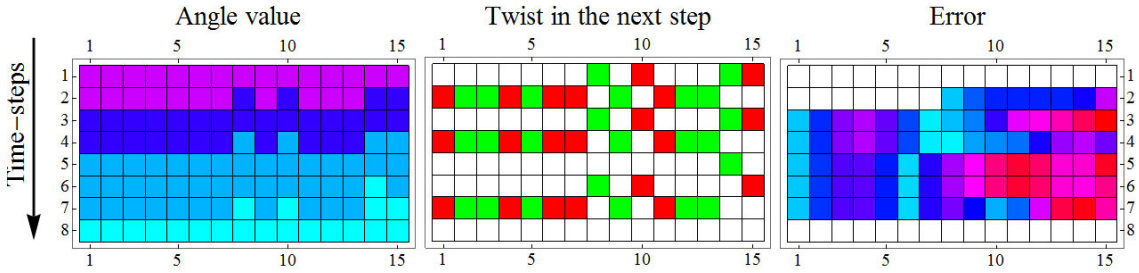

Figure 15: The values of angular parameters and errors for each module in AZ during the entire transition produced by PSO. The color convention as in Figure 7. The maximal error is $\max \delta_y = 0.172$.

into the fully optimal solution in the previous case where $k = 4$. Figures 14 and 15 show: the best result found, and its analysis, respectively ($t_s = 0, t_e = 10$ here and in all the examples below).

As Figures 14 and 15 indicate, the solution found by PSO is much shorter. In other words, it reaches the final configuration in much fewer time-steps. Moreover, this solution results in smaller maximal errors than the greedy algorithm, which is clearly visible in the comparison plot shown in Figure 16.

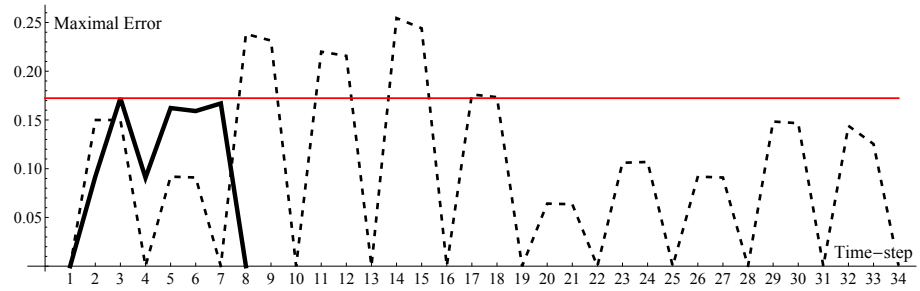

Figure 16: The comparison between the greedy algorithm (black dashed line) and PSO (black solid line). The number of steps and maximal error are substantially smaller in the AZT produced by PSO.

As the figure clearly indicates, the "greedy" approach although intuitive, produces much worse result than PSO. It is also interesting to compare the middle parts of Figures 13 and 15. The modules in the "greedy" solution make a lot of changes, e.g. the last (15^{th}) module makes almost continuously 21 "clicks" to the left, which results in 3.5 turns, which effectively gives rotation of half a turn. The 5^{th} module first twists to the right, then to the left and again to the right. On the contrary, the PSO solution is very "economical", i.e. each module changes position the minimal number of times (3) in one direction only.

4.3 Manipulating under environmental constraints

This example reflects a real-life problem of controlling a manipulator by finding a transition which would bring the tip of the manipulator into a desired position, assuring that environmental constraints are not violated at any time during the transition. The geometrical conditions of the experiment correspond to a physical experiment described in details in [37], and can be summarized as follows:

- There is a 12-unit dodecagonal manipulative in a box.
- The manipulative is fastened vertically to the bottom of the box.
- There is a circular opening in one vertical side of the box.

The challenge presented in [37] was to manually find the angular positions of the modules of the manipulative within the confines of the box so that the tip of the manipulative gets through the circular opening, as illustrated in Figure 17. Figure 18 shows selected time-steps of the manual manipulation of the physical model described in [37].

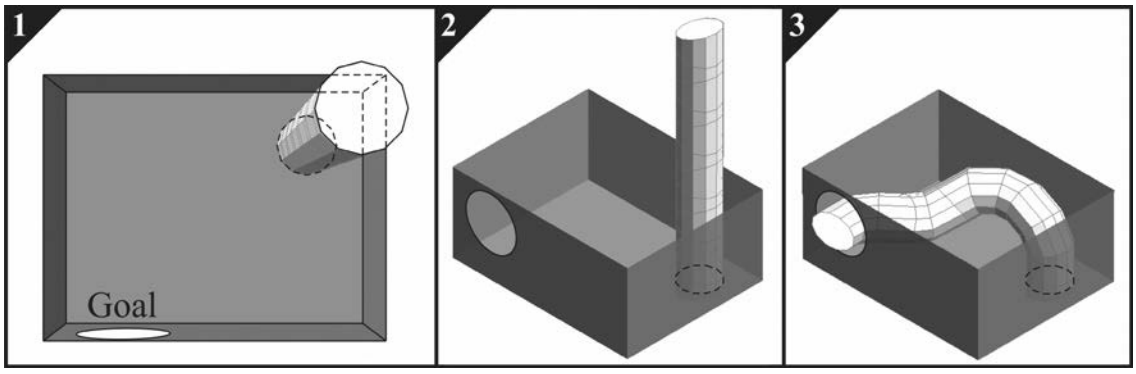

Figure 17: The manual experiment with a 12-unit dodecagonal manipulative. Sub-figures 1 and 2 show: the top and axonometric views of the experimental setup in state $\mathbf{S_s}$, respectively. 3. The axonometric view of $\mathbf{S_e}$ which has been found by the manual manipulation.

After approximately 30 minutes of practicing manipulations of this physical AZ and a few minutes of rehearsal for this particular task, the completion was relatively quick - 2'30". This is in line with [38] which states that adding human kinesthetic sense reduces the errors [39] and the time for completion of the task [40]. The result of this experiment is used as the end configuration $\mathbf{S_e}$ in the next section.

4.3.1 Transition to the given $\mathbf{S_e}$

The problem to be solved by PSO is defined as follows:

- The number of facets of AZ module $n = 12$
- The initial and final states are: $\mathbf{S_s} = [6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6]$ and $\mathbf{S_e} = [6, 0, 0, 9, 2, 11, 7, 2, 7, 7, 1]$. As stated above, $\mathbf{S_e}$ is taken from the manual experiment.
- Simultaneously minimize D, w, o.
- The objective function is $f = o + 0.1 \max w + D$ which foremost assures that all the points lie in the desired operational volume ($o = 0$) and that the destination state is reached ($D = 0$).

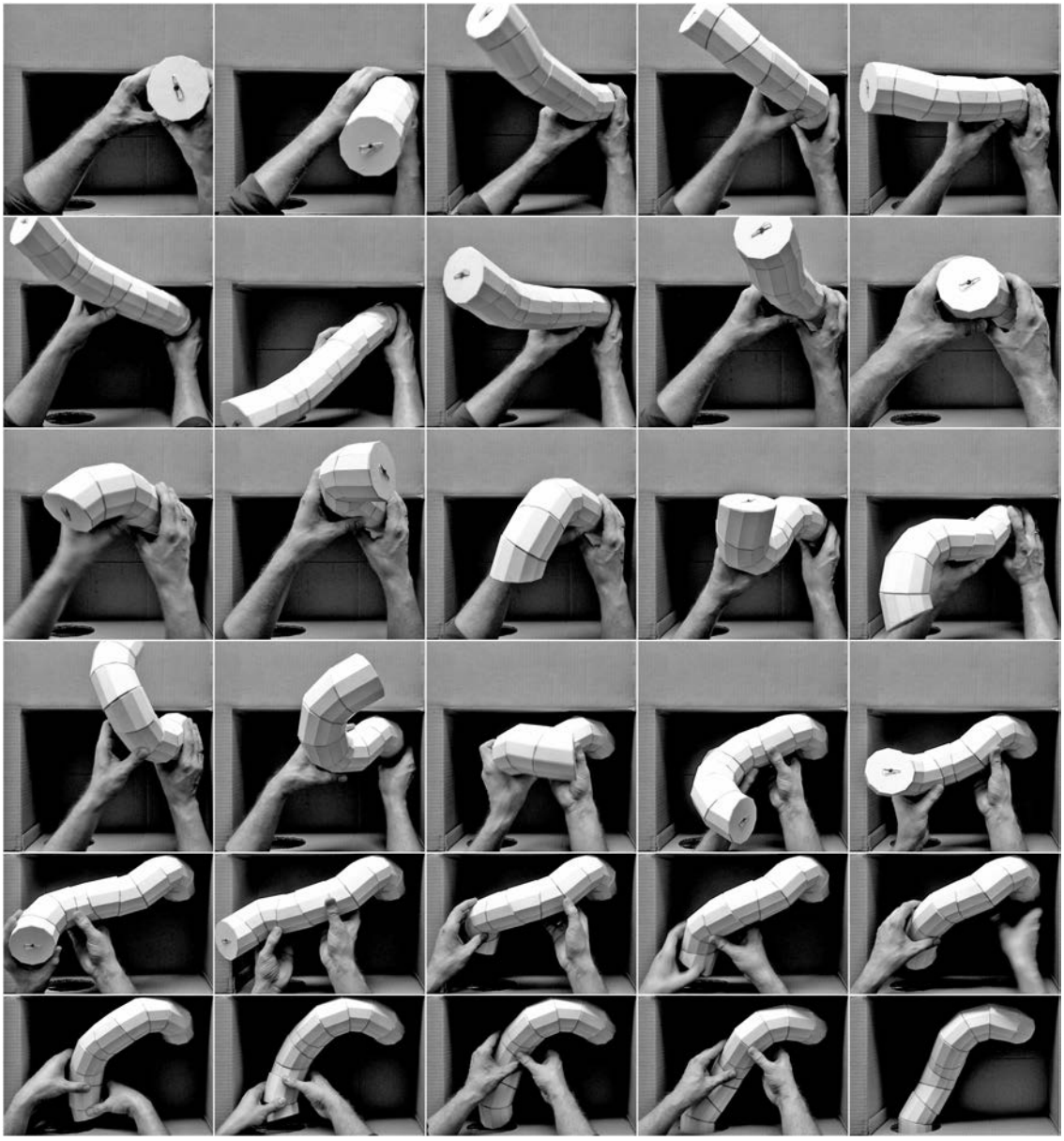

Figure 18: A series of 30 photographs taken at 5 second intervals showing the completion of the task. The modules are 80 mm in diameter and allow for continuous twist rotation, which is advantageous. On the other hand it is very difficult or impossible to manually twist more than one module at the time.

- Additionally, minimize the maximal deviation w of the corresponding positions of the centroids of each module among the consecutive steps. In other words, the motion of AZ to be relatively smooth, which is equivalent to the reduction of wobbling.

The best found values of the objective function are 0.389, 0.275, and 0.302 for $k = 1, 2, 3$ respectively. Unlike in the previous case, this time introducing accelerations $k > 1$ gave significantly better results then for constant angular velocity transformation. Still, the burden associated with more variables for $k = 3$ leads to a worse solution when compared to the simpler case $k = 2$. Figure 19 visualizes the best AZT found by PSO.

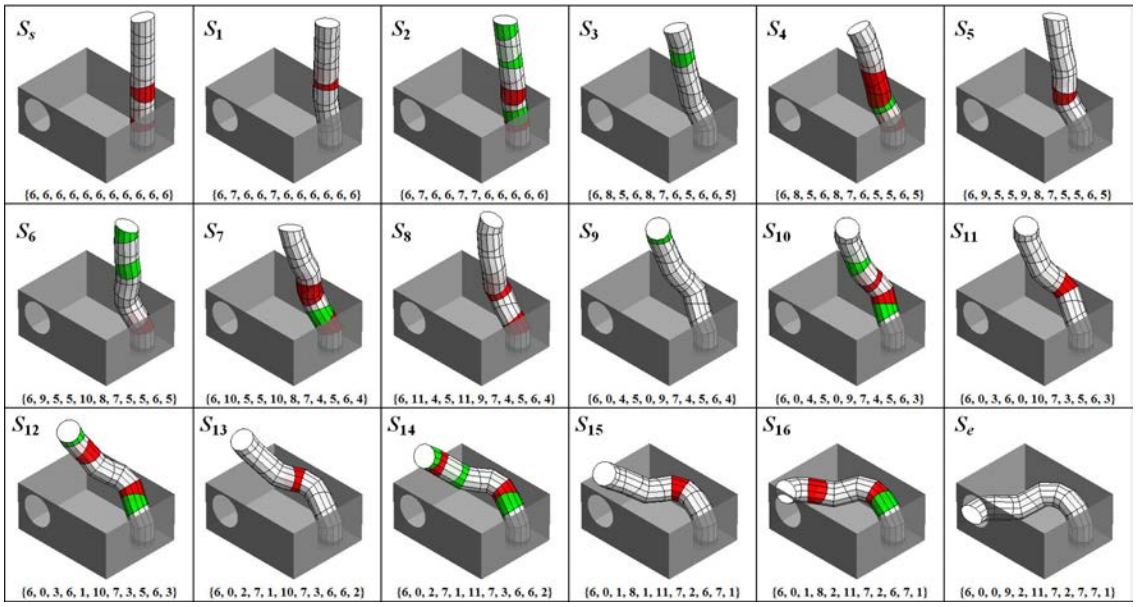

Figure 19: The transition found by PSO requires only 17 time-steps. Green and red indicate the relative twists in: right and left, respectively

4.3.2 Transition to a given point in 3D space

This subsection concerns itself with even more realistic task for a manipulator, since the final state $\mathbf{S_e}$ is assumed to be unknown. The environment is the same as described above, however, the challenge here is to bring the center of last unit, $\mathbf{p_e}$, as close as possible to the desired target point $\mathbf{p_t}$. At the same time the wobbling should be minimized as well. The objective function becomes $f = 3o + 0.1 \max w + d$ (the factor 3 was used in order to give relatively more importance to the environmental constraints; $o = 0$ at the end of optimization process), $d = |\mathbf{p_e} - \mathbf{p_t}|$. $\mathbf{p_t} = (-23.8, -23.5, 10.9)$ which is the position of the final configuration described in the previous section, and in accordance with the geometrical properties stated in Section 2.

Similarly like in the previous case, PSO as a result gives a number of near-optimal solutions with similar final values of the objective function. One of the best is depicted in 20. It brings the center of the last unit to the point $\mathbf{p_e} = (-24.0, -23.5, 10.8)$ giving $d = 0.24$. Optimization defined in this way will always give multiple, similar transformations which will reflect the process of finding the balance between minimizing d and the wobbling w. One can easily give more importance to one or another by introducing desired multiplication factor.

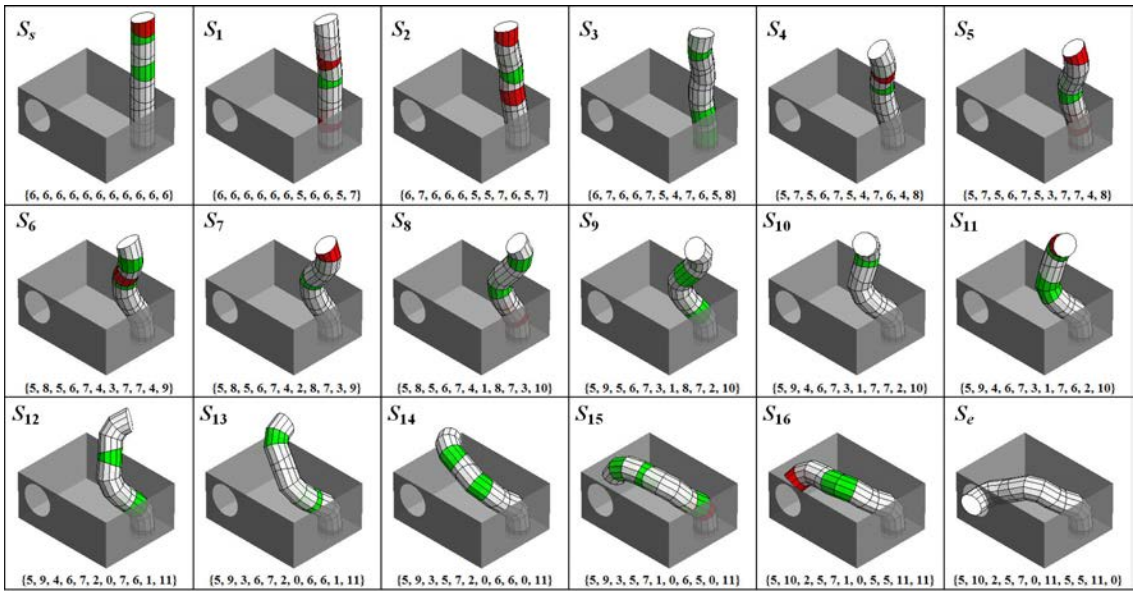

Figure 20: The transition found by PSO requires only 17 time-steps. The color convention as in Figure 19.

5 Conclusions and future work

Arm-Z (AZ) is an Extremely Modular System (EMS), which in principle is very simple, as it is composed of congruent modules. The two major advantages of this approach are: economization (since the identical modules can be mass-produced), and robustness of the system. The latter is due to its hyper-redundant nature: modules which failed can be easily replaced, also if some fail, the manipulator may still perform some desired tasks. However, the disadvantage of all EMSs is that their control is non-intuitive, or simply put - difficult. The approach to AZ control presented here is discrete and highly nonlinear. Therefore the use of combinatorial techniques at a first glance may seem as a potential solution. Unfortunately, due to the "combinatorial explosion" they fail quickly even at relatively small cases. Other intuitive strategies (e.g. presented here greedy algorithm) can produce relatively good solutions, however, they will most likely not be globally optimal. This work demonstrates a successful implementation of a meta-heuristic method, namely Particle Swarm Optimization (PSO) to preliminary control of an AZ manipulator. It has been shown that it can efficiently find transformations, also for quite complex tasks.

Regarding potential applications, it is important to note that due to the nature of AZ, the head unit seems not to have the ability of reaching the entire continuous $\mathbb{R}^3$ space. Detailed discussion regarding this subject is beyond scope of this paper. Nevertheless, clearly the concept can be applied for tasks where great accuracy in finding *any* point in 3D space (obviously within a given range) is not required (e.g. as in the case of *cleaning heads*).

Future research will be focused on the efficiency of PSO for faster computations, including massive parallelization with GPUs. This is a necessary improvement in order to allow for a real-time control. Due to the nature of this problem it is a quite straightforward task. Also, handling self-crossing prohibition (here the AZ was too short for the problem to occur) will be implemented, as well as various kinds of constraints, related to: driving motors, theirs speed, other characteristics, fault recovery etc. Reflecting the physical properties of driving motors, continuous twists rather than discrete will be analyzed for improved efficiency and practicality of AZ.

Acknowledgments

This work was completed as part of the project titled: *"Innovative Extremely Modular Systems for temporary and permanent deployable structures and habitats: development, modeling, evaluation & optimization"*. It was funded by *"Polonez 2"* research grant no. 2016/21/P/ST8/03856 supported by the National Science Centre, Poland. This project has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Skłodowska–Curie grant agreement No 665778, .

References

[1] Machi Zawidzki. *Discrete Optimization in Architecture: Extremely Modular Systems*. SpringerBriefs in Architectural Design and Technology. Springer, 2016.

[2] Machi Zawidzki. Creating Organic Three-dimensional Structures For Pedestrian Traffic with Reconfigurable Modular "Truss-Z" System. *International Journal of Design & Nature and Ecodynamics*, 8(1):61–87, 2013.

[3] Machi Zawidzki and Katsuhiro Nishinari. Modular Truss-Z system for self-supporting skeletal free-form pedestrian networks. *Advances in Engineering Software*, 47(1):147–159, 2012.

[4] Machi Zawidzki and Katsuhiro Nishinari. Modular Pipe-Z system for three-dimensional knots. *Journal for Geometry and Graphics*, 17(1):81–87, 2013.

[5] M. Zawidzki and T. Nagakura. Arm-Z: a modular virtual manipulative. In H-P. Schröcker and M. Husty, editors, *Proceedings of the Sixteenth International Conference on Geometry and Graphics*, Conference Series, pages 75–80, Innsbruck, Austria, 2014. Innsbruck University Press. ISBN 978-3-902936-46-2.

[6] Machi Zawidzki. Deployable Pipe-Z. *Acta Astronautica*, 127:20–30, 2016.

[7] Wilhelm Fuhs and Hellmuth Stachel. Circular pipe-connections. *Computers & Graphics*, 12 (1):53–57, 1988.

[8] James Gray. The mechanism of locomotion in snakes. *Journal of experimental biology*, 23(2): 101–120, 1946.

[9] S Hirose. Biologically inspired robots: Snake-like locomotors and manipulators,(1993).

[10] K. Ning and F. Wrgtter. A novel concept for building a hyper-redundant chain robot. *IEEE Transactions on Robotics*, 25(6):1237–1248, Dec 2009. ISSN 1552-3098.

[11] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[12] M. Rolf and J. J. Steil. Efficient exploratory learning of inverse kinematics on a bionic elephant trunk. *IEEE Transactions on Neural Networks and Learning Systems*, 25(6):1147–1160, June 2014. ISSN 2162-237X.

[13] A. Melingui, C. Escande, N. Benoudjit, R. Merzouki, and J.B. Mbede. Qualitative approach for forward kinematic modeling of a compact bionic handling assistant trunk. *IFAC Proceedings Volumes*, 47(3):9353 – 9358, 2014. ISSN 1474-6670. 19th IFAC World Congress.

[14] V. Falkenhahn, A. Hildebrandt, R. Neumann, and O. Sawodny. Dynamic control of the bionic handling assistant. *IEEE/ASME Transactions on Mechatronics*, 22(1):6–17, Feb 2017. ISSN 1083-4435.

[15] Gregory S Chirikjian and Joel W Burdick. A hyper-redundant manipulator. *IEEE Robotics & Automation Magazine*, 1(4):22–29, 1994.

[16] M. Zawidzki and J. Szklarski. Preliminary optimization of Pipe-Z reconfiguration. In G. Várady P. Iványi, B.H.V. Topping, editor, *Proceedings of the Fifth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, 1759-3433, pages 1–12, Stirlingshire, UK, 2017. Civil-Comp Press.

[17] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286 – 300, 2015. ISSN 1568-4946.

[18] Machi Zawidzki. Dynamic shading of a building envelope based on rotating polarized film system controlled by one-dimensional cellular automata in regular tessellations (triangular, square and hexagonal). *Advanced Engineering Informatics*, 29(1):87–100, 2015.

[19] Machi Zawidzki and Katsuhiro Nishinari. Application of evolutionary algorithms for optimum layout of Truss-Z linkage in an environment with obstacles. *Advances in Engineering Software*, 65:43–59, 2013.

[20] Machi Zawidzki. Optimization of multi-branch Truss-Z based on evolution strategy. *Advances in Engineering Software*, 100:113–125, 2016.

[21] Machi Zawidzki. Retrofitting of pedestrian overpass by Truss-Z modular systems using graph-theory approach. *Advances in Engineering Software*, 81:41–49, 2015.

[22] James Kennedy and Russell Eberhart. Particle swarm optimization. volume 4, pages 1942–1948, 1995. cited By 28006.

[23] D. K. Lobiyal Pawan Kumar Tiwari Abdul Hanan Abdullah Omprakash Kaiwartya, Sushil Kumar and Ahmed Nazar Hassan. Multiobjective dynamic vehicle routing problem and time seed based solution using particle swarm optimization. *Journal of Sensors*, 2015 (Article ID 189832), 2015.

[24] Jacek Szklarski and Marcin Wikło. Designing of elastoplastic adaptive truss structures with the use of particle swarm optimization. *Mathematical Problems in Engineering*, 2015 (Article ID 652824), 2015.

[25] Yudong Zhang, Shuihua Wang, Genlin Ji, and Zhengchao Dong. An MR brain images classifier system via particle swarm optimization and kernel support vector machine. *The Scientific World Journal*, 2013, 2013.

[26] Houxian Zhang and Zhaolan Yang. Large-scale network plan optimization using improved particle swarm optimization algorithm. *Mathematical Problems in Engineering*, 2017, 2017.

[27] Yudong Zhang, Shuihua Wang, and Genlin Ji. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015, 2015.

[28] M. Clerc. *Particle Swarm Optimization*. ISTE. Wiley, 2010. ISBN 9780470394434.

[29] Mohammad Reza Bonyadi and Zbigniew Michalewicz. Particle swarm optimization for single objective continuous space problems: A review. *Evolutionary Computation*, 25(1):1–54, 2017.

[30] Alireza Alfi and Hamidreza Modares. System identification and control using adaptive particle swarm optimization. *Applied Mathematical Modelling*, 35(3):1210 – 1221, 2011. ISSN 0307-904X.

[31] Wei-Der Chang and Shun-Peng Shih. PID controller design of nonlinear systems using an improved particle swarm optimization approach. *Communications in Nonlinear Science and Numerical Simulation*, 15(11):3632 – 3639, 2010. ISSN 1007-5704.

[32] Vijay Kalivarapu and Eliot Winer. A study of graphics hardware accelerated particle swarm optimization with digital pheromones. *Structural and Multidisciplinary Optimization*, 51(6): 1281–1304, Jun 2015. ISSN 1615-1488.

[33] Jitendra Kumar, Lotika Singh, and Sandeep Paul. GPU based parallel cooperative particle swarm optimization using C-CUDA: a case study. In *Fuzzy Systems (FUZZ), 2013 IEEE International Conference on*, pages 1–8. IEEE, 2013.

[34] Yuhui Shi and RussellC. Eberhart. Parameter selection in particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII*, volume 1447, chapter Lecture Notes in Computer Science, pages 591–600. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-64891-8.

[35] Wang Jin. Particle swarm optimization with adaptive parameter control and opposition. *Journal of Computational Information Systems*, 7(12):4463–4470, 2011.

[36] Machi Zawidzki. Arm-Z Manipulations, 2014. URL `http://demonstrations.wolfram.com/ArmManipulations/`. An interactive demonstration.

[37] Machi Zawidzki. Pipe-Z virtual and physical manipulatives. *Virtual Reality*, 2017. (under review).

[38] M Osama Alhalabi and Susumu Horiguchi. Haptic cooperative virtual workspace: Architecture and evaluation. *Virtual Reality*, 5(3):160–168, 2000.

[39] Ian Oakley, Marilyn Rose McGee, Stephen Brewster, and Philip Gray. Putting the feel inlook and feel . In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 415–422. ACM, 2000.

[40] Miyasato Noma and Tsutomu Miyasato. Cooperative object manipulation in virtual space using virtual physics. *Proceeding of Dynamic System and Control ASME*, 61:101–106, 1997.