EXPERIMENTS WITH A FOLDING MULTI-AGENT SYSTEM IN THE DESIGN OF TRIANGLE MESH STRUCTURES

Benjamin FELBRICH[1], Daniel LORDICK[2], Jörg Rainer NOENNIG[1] and Sebastian WIESENHUETTER[1]
[1]TU Dresden, Germany, Dept. of Architecture, Juniorprofessorship of Knowledge Architecture
[2]TU Dresden, Dept. of Mathematics and Science, Institute of Geometry

ABSTRACT: In the presented work we attempt to find a way to utilize the collaborative intelligence of bio-inspired multi-agent systems for an architectural application. To do so we establish, geometrically describe, optimize and program a physical object. Our setup is a foldable structure of interconnected triangles where simple robotic agents control the angles between adjacent facets. This structure has the capacity to reshape itself in a self-directed manner. The preliminary objective was to let the structure approximate a certain target shape. Beyond this we investigate more general concepts for the application of the system. Apart from the geometrical foundation and the computer simulation of the folding behavior, the actual prototyping with physical models plays a significant role.

Keywords: prototyping, rigid folding, triangle mesh, free form discretization

1. INTRODUCTION

In nature, systems consisting of a large number of simple individuals achieve stunning goals by interacting in both physical and cognitive ways. Both ways, from collaboratively lifting heavy objects to the optimization of food paths, have in common that the abilities, which emerge from the group, go far beyond the abilities of an individual actor ("agent").

We tried to find a mechanism resp. construction kit which utilizes the logic of emergence for a defined architectural purpose – the erection of free form structures. Composed from a multitude of simple robotic agents, the structure is supposed to reshape itself and thereby approximately regenerate an arbitrarily chosen free form surface. Such a device would be able to directly construct a form imagined by the architect, or to generate a form which is optimal under certain conditions. Furthermore it could be used as a design tool, which gives a direct physical feedback to form alterations. In larger scale applications such robot driven structures could serve as a flexible concrete formwork: after assembling a particular configuration of triangular agents, a set of free form shapes within a certain design space can be erected.

Upon close investigation, a folding mechanism appeared promising, in which each folding edge is controlled by a single agent and all interconnected parts form a large dynamic system, a discrete surface. The distribution and the alignment of the agents were the first important issue to be closely investigated. This topic belongs to the realm of free form discretization and optimization, since the mechanism's underlying folding pattern is derived from the triangulation of the target surface to regenerate. The other important part of the project was to elaborate an agent design that is capable of lifting weight by rotating adjacent frames. It was developed with the help of physical prototypes, which were tested, evaluated and enhanced step by step.

In the following, both parts of the work will be shown. In the end we will present an experimental setup in which the device has to reach a predefined target surface. Prior to this, some preconditions need to be clarified.

2. ASSUMPTIONS

Our approach necessitates some characteristics which correlate with cost-relevant requirements of real-world applications like standardization. Beside this positive aspect the system nevertheless should offer a rather broad range of achievable shapes. In detail we make the following assumptions to assure feasibility:

- **Simplicity** - the system and its parts have to be as simple as possible
- **Similarity** - each robotic agent has to resemble its neighbor
- **Interdependence** – to reach the common goal, each agent has to actively participate
- **Smoothness** - the triangulated structure has to be as smooth and as close to the target surface as possible
- **Flexibility** - the system has to have the ability to approximate any free form surface

To achieve the latter the structure is made of a set of triangular parts with a limited variety. The configuration of the triangles can be changed manually to access different design spaces.

3. DEVELOPING PROCESS

The consecutive steps in the process of developing the structure and the algorithms to control the structure, were as follows: starting from the design of a free form surface, the mechanism was constructed to approximate this designed target surface. Then the mechanism was folded into an arbitrary state. From this state it had to restore the initial constitution by itself.
After this goal was reached, the target surface can be changed. Now the construction kit can be reconfigured and thus be adapted to the new conditions – without changing the agents’ design but only their topological alignment.

4. TRIANGULATION

The abstraction of the described folding setup is represented by a triangular polygon mesh, in which the folding agents are located on the inner mesh edges.

Since we intend to establish a construction kit, which is affordable and easy to produce, we have to maximise the degree of repetition of components the system is made of. A conventional free form triangulation, which in general produces nothing but unique edge lengths, would have forced us to individually draw and produce every piece of the frame. Though this process can be handled by means of parametric modelling and tools like computer aided manufacturing (CAM), the re-configurability would be lost.

A more feasible approach was found in limiting the amount of utilized edge lengths in the triangulation to a relatively low number. Thus a limited set of combinable triangles emerges that can be easily reproduced.

In order to approximate any free form surface we had to investigate the relation between smoothness and standardization and derive an algorithm to triangulate free form surfaces using a limited amount of edge lengths. It is inherent to this problem that we are not able to achieve a triangulation result in which every vertex of the mesh is placed on the target surface S or in which every edge precisely equals an item of the limited set of edge lengths yet certain deviations may be accepted in this regard.

4.1 Discretization – Standardization versus Smoothness

It is save to say that complexity C increases with the number of edge lengths utilized in the triangulation. C needs to be minimized. Smoothness on the other hand has a close relation to the discrete Gaussian curvature. In [1] Alexander Bobenko defined the discrete Gaussian curvature K in a node p of a discrete surface S as:

$$K(p) := 2\pi - \sum_i \alpha_i$$

Here α_i are the angles between nearby edges adjacent to p.

In the simplest setup we can assume all edge lengths to be equal and thus form only equilateral triangles. This setup provides the highest degree of repetition, i.e. C cannot be lowered any further. In this case we have distinct steps of discrete Gaussian curvature depending on the number of edges adjacent to p. A node p with n adjacent edges will have the curvature:

$$K(p) = 2\pi - n \cdot \frac{\pi}{3}$$

This results in a very limited variety of curvature that can be obtained with one edge length available. We consider the number of alternatives for the discrete Gaussian curvature in each node to be the graduator for the obtainable smoothness in the triangulation. In order to increase this smoothness but still keep a high degree of repetition in the utilized pieces, we allowed the triangulation algorithm to use a limited set of edge lengths larger than one. Doing so, the number of alternatives of discrete Gaussian curvature per node increases exponentially with the number of available edge lengths.

To allow the system to move more freely, we perforated the mesh by removing certain edges. Physical mockups showed that a favorable folding behavior can be achieved if every node of the mesh is either a border node itself, or if it is neighbor to at least one border node.
As abbreviation, the described mesh will be called a "Perforated Limited Triangular Mesh" (PLT-Mesh) with a rank $|L|$, in which $|L|$ describes the number of different edge lengths used. Figure 1 shows an example of a PLT-Mesh of the rank 4.

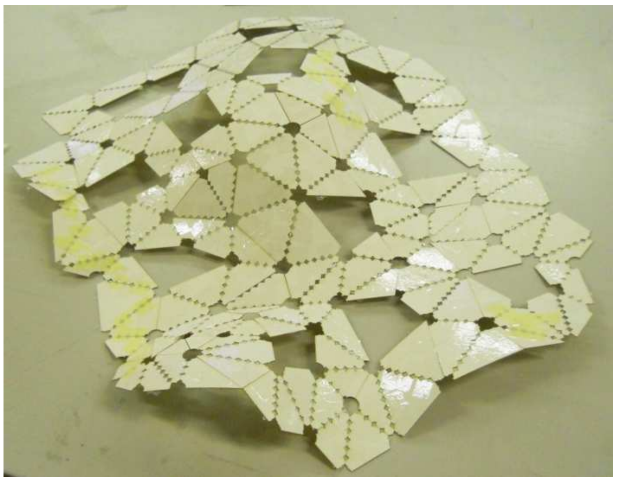
Figure 1: PLT-Mesh of 4th rank

4.2 Triangulation algorithm

To harness the theoretical potential of the limited mesh topology in smoothness and standardization, it was essential to find a way to properly discretize the given free form surface S only using l_i. The algorithm established for that purpose is as follows:

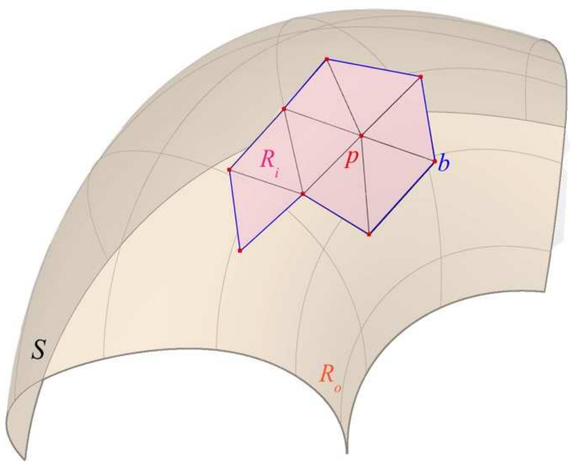

Figure 2: Definitions

Definitions:

S Surface to triangulate

L	a finite set of previously chosen values for edge lengths with $\lvert L \rvert$ being its cardinality
l_i	the elements of L
p	nodes of the polygon mesh to be compiled
b	boundary of the polygon mesh, the closed polyline consisting of all the mesh edges with only one adjacent mesh face
b_p	normal projection of b on S
R_i	the region of S that is inside b_p
R_o	the region of S that is outside b_p

Basic algorithmic procedure:

1. Set one initial triangle arbitrarily with the condition that all of its three vertices are placed on S and only l_i are used as edge lengths.
2. Choose one of the triangle's edges and find two new lines (again only using l_i) to form a new neighbor triangle on S. (by creating spheres with the radii l_i around the respective nodes. One of the two intersection points x_1 and x_2 of these spheres and S is the new node.

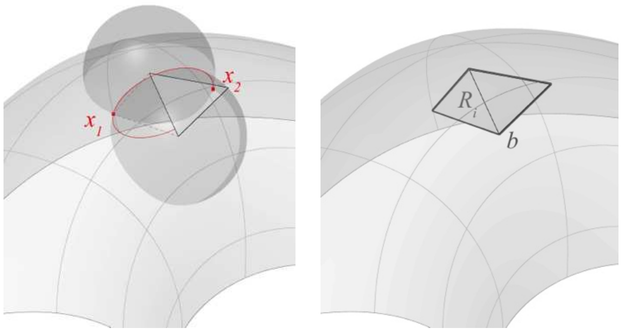

Figure 3: Steps two and three of the triangulation algorithm

3. Include the new triangle and update the polygon mesh and b resp. b_p
4. Examine b and pick a single line segment or two adjoining segments of b to operate on. Then pick and execute one of the following options[*]:

4.a) Repeat step 2 with a single segment of b (which adds one new triangle).

4.b) If two adjoining line segments of b are chosen, fill the narrow gap they span over R_o with two new triangles. This can be geometrically solved by creating $\lvert L \rvert$ spheres with radii l_i around each of the three nodes of the chosen polyline segment. The intersection points of these spheres form a point cloud in which every point has the property of correct distances from the adjacent nodes. From this cloud of points, pick the one with the shortest distance to its normal projection on S and with its normal projection on S being outside R_i. This point represents the new node to add two triangles.

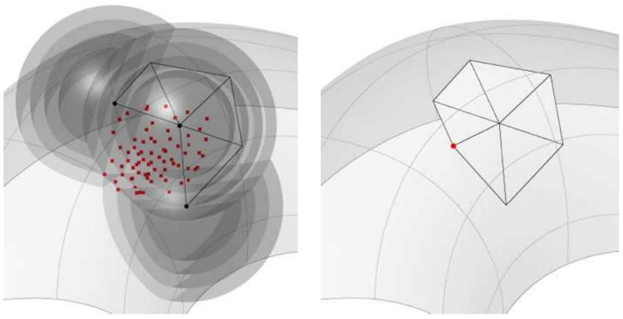

Figure 4: Step 4.b) of the triangulation algorithm

4.c) If the angle between two neighbor segments over R_o falls below a certain threshold value, directly connect the segments' two outer nodes with a new line.

5. If S is filled with triangles, stop. Otherwise go back to 3.

[*] The conditions under which the operation mode in 4. is chosen represents a complex issue of optimizing the algorithm's performance. The priority is to find the narrowest angle over R_o between adjoining segments of b and close them using mode 4.b) preferably or 4.c) in critical situation. These conditions are inspired by and similar to the behavior that wasps show when choosing the most enclosed comb cell to fill when building their nests [2]; it is crucial for avoiding self intersections in b_p and make the triangles grow evenly to the outside over S.

The described algorithm was coded and executed using Grasshopper in Rhino 5. It produces inevitable inaccuracies since it partly draws new nodes near *S* but not necessarily places them on *S*. Thus the resulting mesh is a rather bumpy approximation of *S*. This geometrically inherent error can be partly or completely assigned to the accuracy of the edge lengths by replacing them with their normal projection on *S* (very inaccurate) or pulling them towards *S* using a solver implemented in the particle spring engine Kangaroo (more accurate). Thus it is possible to gradually increase smoothness if a certain tolerance in edge length deviation is allowed.

With this algorithm we found a way to optimally place the individual components of the multi-agent system on a free form.

5. PROTOTYPING

The following paragraphs describe, how the agents were designed and how the prototypical test structure was executed. Looking for different ways to empower the agents, a mechanism with a spindle drive appeared most suitable, in which a rotating shaft pushes or pulls pieces of the frame apart or towards each other. This option was chosen due to its light weight, its low cost and the favorable power transmission. Figure 5 shows the basic idea; figure 6 the section of the frames indicating the most important parameters.

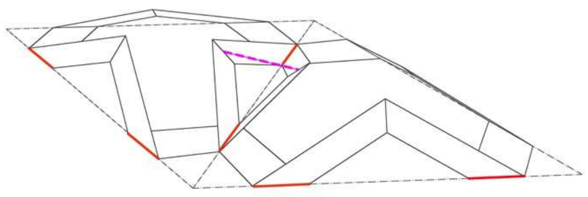

Figure 5: Basic idea; red: hinges, magenta: shaft to push / pull the frame

After determining the agent's basic design in drawings and simple mockups, three consecutive prototypes (so-called "Kingfisher") were elaborated. To quickly proceed in their evolution, small groups of agents (up to three) were built, tested, and improved in regards to their strength and robustness. This method of operation represented an important shift in our approach from a merely theoretical, simulation-based mode to the very practical design-oriented model making. It was inspired by the shifts in innovation emergence in a design-context described in [3].

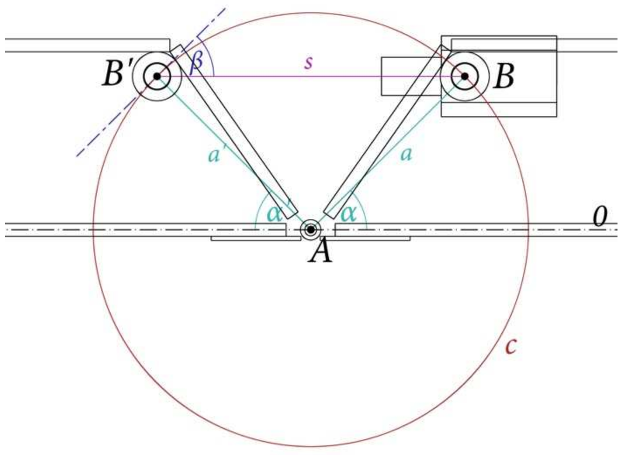

Figure 6: Basic section

5.1 Kingfisher 0.1

The very first prototype (figure 7) consisted of a MDF frame, metallic hinges and was empowered with a 12V Motraxx X-Slot Race 143 model train motor with an idle speed of about 22000 rpm and a torque of 1.3 Nmm (manufacturer information). It was controlled using an Arduino Uno and the Firefly plug-in for Grasshopper in Rhino 5. The major insight from this model was that the utilized MDF was to heavy for the relatively weak motor. Furthermore it revealed the major importance of the shield angle α (see figure 6), since it directly affects the leverage and therefore the power transition from the motors rotation force into a push / pull force along the threaded shaft *s* into a rotation of the frames about their common edge *A*.

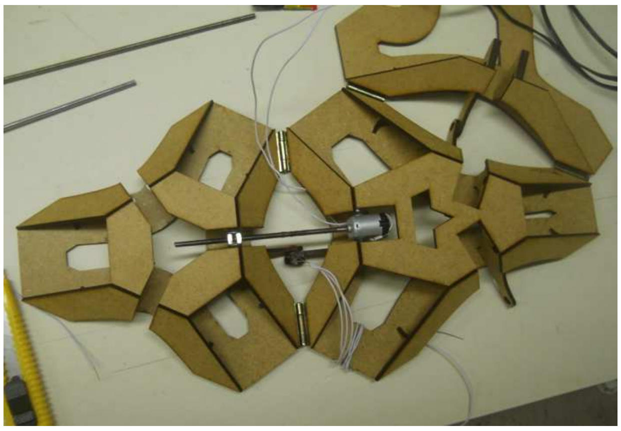
Figure 7: Kingfisher 0.1

5.2 Kingfisher 0.2

In the second prototype shown in figure 8 the shield angle α was optimized to improve the agility by a better power transmission. Further, a lighter foam board material and the stronger motor “Motraxx X-Slot 283 Tuning” (4.0 Nmm torque) were utilized.

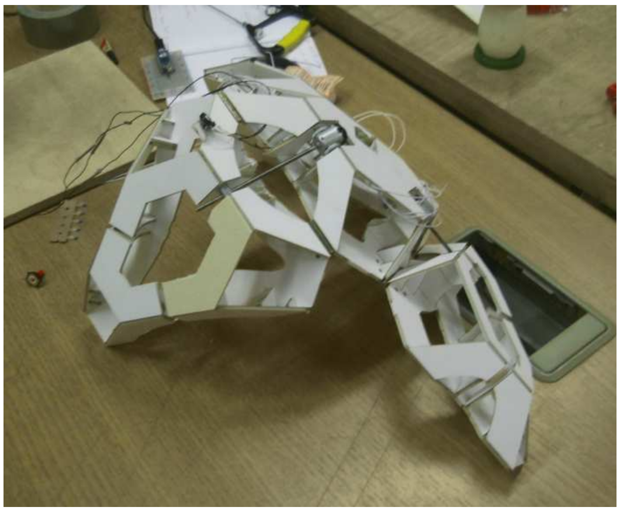
Figure 8: Kingfisher 0.2

The mount of the motors and their counter pieces were improved just like the overall precision of the model making. Like in its predecessor, the motors were controlled using an Arduino Duemilenove and the interface “Firefly” for Grasshopper in Rhino 5. Furthermore a motor control unit based on the L293 chip was used with an external power source to turn the relatively low 5V control voltage of the Arduino board into a powerful 12V operating voltage with a current of up to 500mA per channel to empower up to two motors. In contrary to its predecessor this prototype was able to lift itself off the ground and showed a much higher ability to move in general.

5.3 Kingfisher 1

In the third prototype (figures 9 to 13) major improvements were made: the empowering unit was separated from the triangular frame which interconnects the agents. This was done to ensure its re-configurability (see section “2. Assumptions”). It is furthermore equipped with a gear box with a transition ratio of 1:4 (figure 9), a stronger motor (“Johnson 20543” with 130 Nmm torque) and a potentiometer mounted to the frames giving a feedback of the rotation and angular state in the agent. Its highly stressed parts (e.g. the mount of the motors) are made of stable MDF while the rest is made of lighter foam board.

With a shield angle α of 47.6° the agent is able to rotate its counter part 62,5° in the upper direction until the agent is fully closed (figure 12). It can also rotate its counter part 62.5° in the lower direction until the operating angle between shaft and the rotation circle’s tangent equals 75°. In this fully opened state (figure 13), about 0.96 times the motor’s force is pushing and pulling the opposite pieces apart instead of making them rotate about *A*. This value was estimated as the upper bound for stress in the frame in this direction, known from the previous models. To prevent the frames from cracks, these values were embedded in the code for the motor not to exceed this area of operation.

In this prototype we used motor control shields based on the L298N chip to receive a powerful operating voltage of 12V and up to 2A current per channel (two channels).

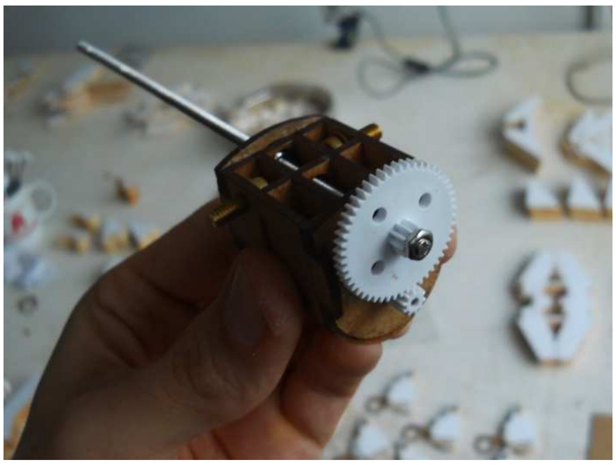
Figure 9: Gear box of Kingfisher 1

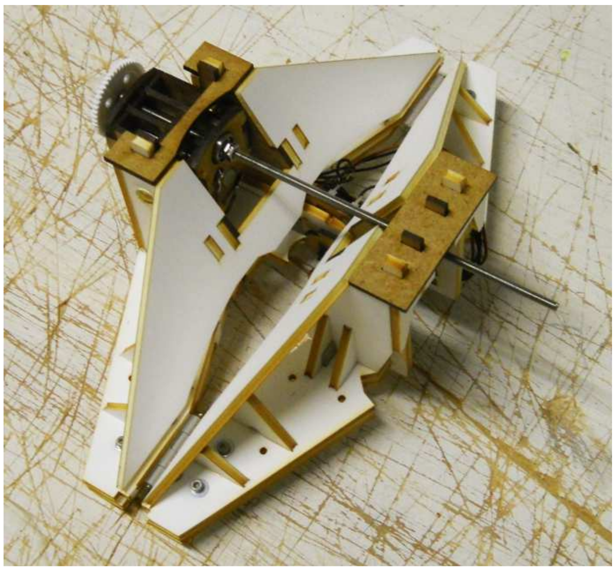
Figure 10: Kingfisher 1, single agent

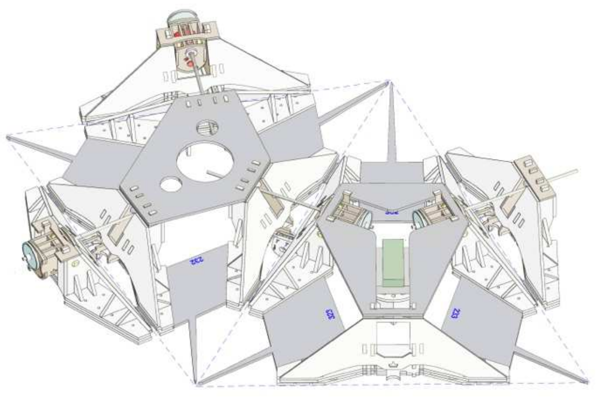
Figure 11: Four agents mounted to connecting triangle frames

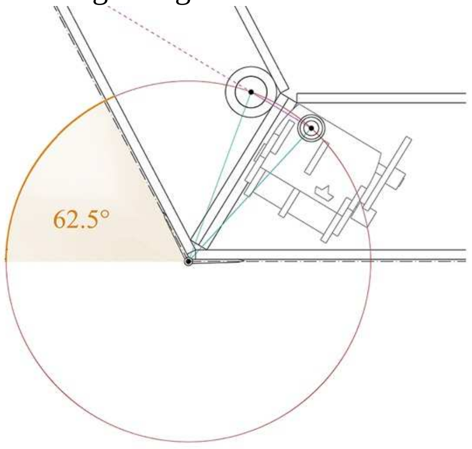

Figure 12: closed state

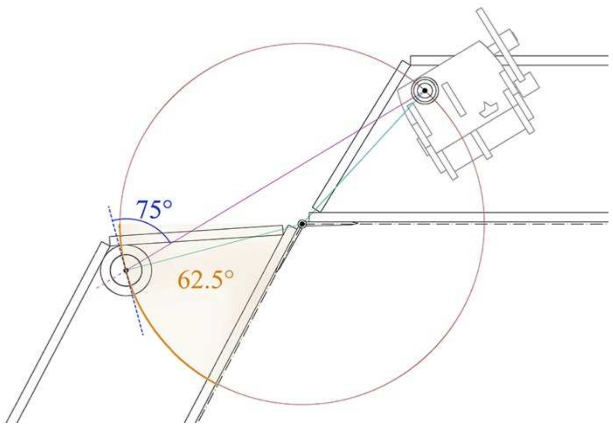

Figure 13: opened state

6. EXPERIMENTAL SETUP

The experimental setup represents the synthesis of a singular agent's design and the logics of the connectivity of a set of agents. It was chosen big enough to provide a sufficiently number of agents – to give a reliable feedback over constraints given by the dimensions. On the other hand it had to be affordable and constructable within only six weeks.

A free form surface was designed in Rhino 3D and triangulated with the algorithm described in section 4.2. Thereupon its appearance could be changed through a simulated folding process using Grasshopper in Rhino 5 and the physics engine Kangaroo. Figures 14 a) to d) show the initial target surface (a), its discretization into a PLT-Mesh of 3rd rank in b) and another state into which it was transformed through folding in c), whereas d) shows the physical model in this state. There are 22 inner edges of which 21 are equipped with agents. The prototype's dimensions are about 1.10m by 1.40m and a height of about 0.50m

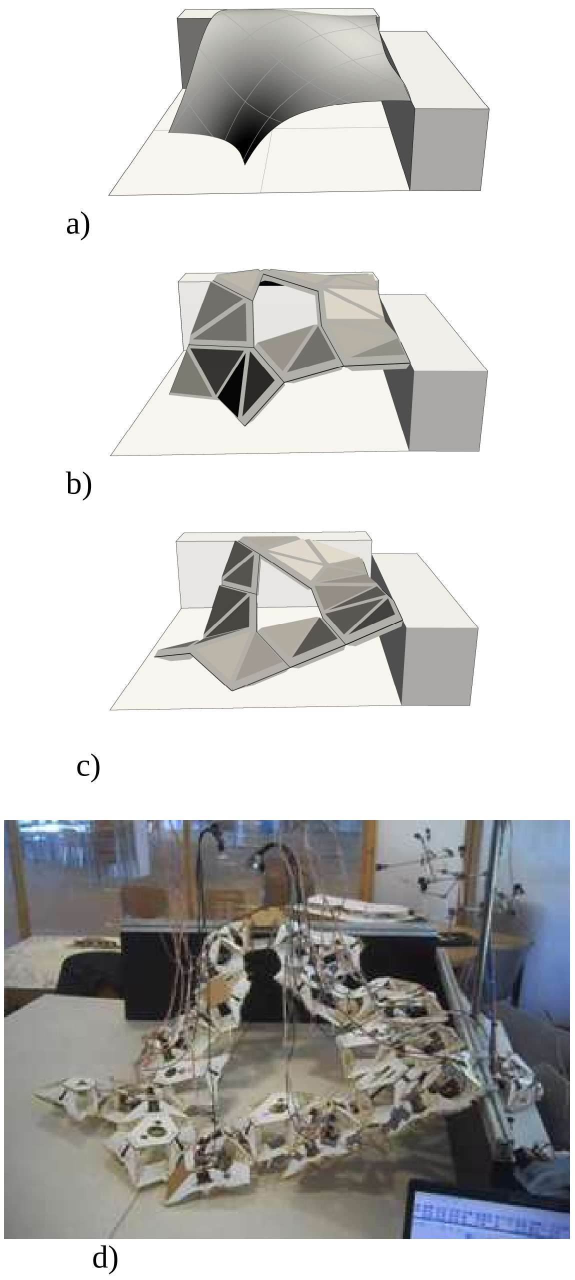

Figure 14: Physical model and test setup

6.1 Electronic Setup

motors: 21 x Johnson 20543, Murata Z5U-5 ceramic capacitor 1 μF (21 times)
motor driver: 12 x L298N two channels, up to 2A each
potentiometer: 21 x Piher PT-15 10 kΩ
control: 1 x Arduino Duemilenove (for global control), 11x SparkFun Pro Micro 5V 16MHz
power supply: 2x LPK2-23 400W power supply, 12V 22A each

One laptop is used to send global ON/OFF and timing signals via Firefly to the Arduino Duemilenove. Then these signals are distributed to the smaller agents' boards SparkFun Pro Micro.

6.2 Coding

While the Duemilenove runs with the Firefly Firmata, the SparkFun Pro Micro boards were equipped with a custom code which utilized the feedback of the potentiometers and global time and switch signals.

In order to reduce cost, not every agent was equipped with one control board. Instead, one control board controls two motors. In the code, however, their circuits run completely independent mimicking an independent behavior of two agents.

The goal of the mechanism was set for it to reach its target shape (see figure 14 b) from any starting position (e.g. figure 14 c). To do so every agent has to reach its individual target angle value which was implemented in the code. When every agent has reached this value, the connected mechanism has the state of the target surface.
Two different controls were tested:

A. Straight forward approach:

1. Compare the potentiometers value to your individual target value;
2. If it doesn't equal, rotate the motor towards the target value

3. If you have reached the target value, stop;

Since the interconnected agents naturally block each other with a simple control mode like this, the system would get stuck and even crack. Previous simulations of the folding process predicted these problems. To avoid this, a more sophisticated simulation was run in which the starting position was well defined, the target surface was reached through folding and no deadlocks and cracks appeared. From this simulation the progressive graph of the angle over each inner edge was recorded. Thus, individual control curves for each agent could be established, translated into an array of 10-bit integers and implemented in their individual codes. Following these individual control curves, the agents would precisely repeat the folding simulation in which they do not block each other and finally reach their goal. For this setup a synchronization through a global timing signal was necessary. The simplified commands look like this:

1. Go to the very first value in your control curve (meaning: reset to starting position)
2. If a certain time interval is exceeded, move to the next value in the control curve.
3. Move towards this control curve value.
4. Go to 2.

Here the choice of the time interval represents an issue of fine tuning.

6.3 Test Results

The two different code setups cause very different outcomes. While the straight forward approach makes the mechanism move quickly but uncoordinated and is also more likely to cause cracks (it was more time spent repairing the model than testing it) the second approach led to a more diffident but relatively weak movement.

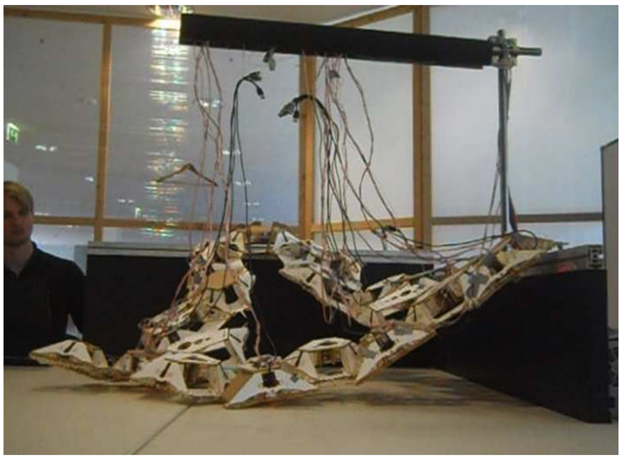

Figure 15: lower state

In both cases the robot was not able to completely reach its goal due to a lack of strength when lifting heavy parts. Even worse, instead of completely lifting itself into its desired position, it cracked parts of the frame. Figure 15 shows its lower, most unstressed position (resp. figure 14 c) while figure 16 shows its best attempt to reach the target position (resp. figure 14 b) achieved with the first code setup (straight forward). Despite this failure the robot shows a very agile movement within its range of abilities.

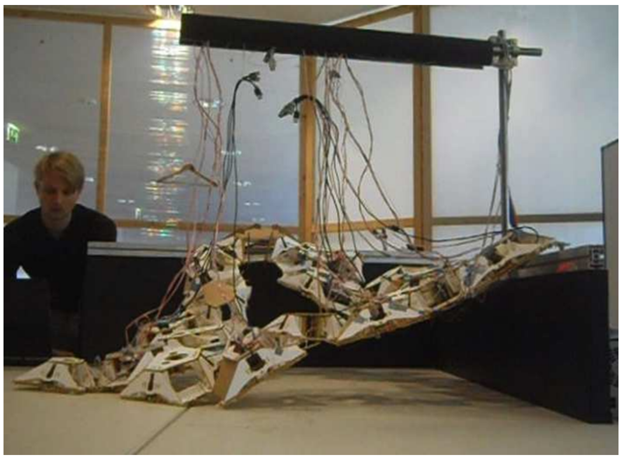

Figure 16: upper state

7. CONCLUSION

The initial goal for the device was to reach a previously defined free form target surface from any initial state and to switch between those states. By way of correct alignment of the pieces in a triangular arrangement it was ensured that – geometrically – the mechanism was able to perform the move. Furthermore the simulation of the desired folding behavior was relatively easy to carry out. In its physical representation, however, the robot failed to reach its goal. This twofold result reveals an advantage of our approach to use both computer aided design and simulation plus physical prototyping for discovering the differences and difficulties that distinguish real world application from its theoretical foundations.

The lack of strength and stability does not only call for stronger frames and motors, but for a more sophisticated and intelligent control mode, which would enable the robot to reach its goal even with limited physical abilities. Such a control, based on agent communication and collaboration would correspond to the natural model of a multi-agent system and be a truly bio-inspired intelligent device.
From this point of view, the mechanism presented can act as a test setup for different codes. It provides a physically limited body, comprised of independent entities, that serves as a nucleus for the testing of the performance of collaborative codes.

In extrapolation of the potentials of the mechanism we see a structure that in future can respond to physical interaction with optimization routines and thus can be a sophisticated raw material for design studies. This may push the idea of prototyping within the realm of architecture onto the next level.

REFERENCES

[1] Alexander I. Bobenko, Yuri B. Suris, “Discrete Differential Geometry: Integrable Structure”, Graduate Studies in Mathematics, Vol. 98, AMS, 2008, page 27-28

[2] Eric Bonabeau, Marco Dorigo, Guy Theraulaz, “Swarm Intelligence - From Natural to Artificial Systems”, A volume in the Santa Fe Institute Studies in the sciences of complexity, 1999, Oxford University Press, page 14-23

[3] Julian Adenauer, Jörg Petruschat: Prototype! - physical, virtual, hybrid, smart - tackling new challenges in design and engineering, form+zweck, 2012, page 12 - 37

ABOUT THE AUTHORS

1. Benjamin Felbrich graduated from the Department of Architecture of TU Dresden. He focuses on the improvements of architectural design through computation and optimization.
2. Daniel Lordick is Professor for Geometric Modelling and Visualization at the Department of Mathematics and Science of TU Dresden, Germany.
3. Jörg Rainer Noennig is Juniorprofessor for Knowledge Architecture at the School of Civil and Environmental Engineering of TU Dresden, Germany.
4. Sebastian Wiesenhütter is an architecture graduate and currently works as a research associate at the chair of Knowledge Architecture at TU Dresden. He focuses on bio-inspired architecture and generative approaches in design.