Open access • Book Chapter • DOI:10.1007/978-3-319-08864-8_21

Programmable Self-assembly with Chained Soft Cells: An Algorithm to Fold into 2-D Shapes — Source link

Jürg Markus Germann, Joshua E. Auerbach, Dario Floreano

Institutions: École Polytechnique Fédérale de Lausanne

Published on: 22 Jul 2014 - Simulation of Adaptive Behavior

Topics: Soft robotics

Programmable self-assembly with chained soft cells: an algorithm to fold into 2-D shapes

Jürg Germann, Joshua Auerbach, and Dario Floreano

Laboratory of Intelligent Systems, EPFL - IMT - STI - LIS
Station 11, 1015 Lausanne, Switzerland
{jurg.germann,joshua.auerbach,dario.floreano}@epfl.ch
http://lis.epfl.ch

Abstract. *Programmable self-assembly of chained modules holds potential for the automatic shape formation of morphologically adapted robots. However, current systems are limited to modules of uniform rigidity, which restricts the range of obtainable morphologies and thus the functionalities of the system. To address these challenges, we previously introduced "soft cells" as modules that can obtain different mechanical softness pre-setting. We showed that such a system can obtain a higher diversity of morphologies compared to state-of-the-art systems and we illustrated the system's potential by demonstrating the self-assembly of complex morphologies. In this paper, we extend our previous work and present an automatic method that exploits our system's capabilities in order to find a linear chain of soft cells that self-folds into a target 2-D shape.*

Keywords: Self-Assembly, Soft Robotics, Modular Robotics

1 Introduction

Modular robots have the potential to adapt their morphologies to achieve desired behaviors upon command [1]. Compared to fixed-morphology robots, modular robots offer greater adaptability and versatility. However, a major challenge for modular robots is to have constituent modules interact in such a way that they self-assemble into a target morphology.

Programmable self-assembly of chained modules is a promising strategy for automatically generating complex robot morphologies. Inspired by protein-folding, it provides a means of constructing morphologically and functionally diverse structures from a minimal set of building blocks. This method has several benefits. First, it is superior to other self-assembly strategies in terms of yield [1] and the range of achievable morphologies [2, 3]. Second, modules can be relatively simple because there is a permanent connectivity constraint between all modules, which reduces hardware complexity (e.g. for communication or attachment/detachment between modules) [4]. Third, because of the system's intrinsic modularity, it is especially well-suited for in-silico optimization of morphology and functionality using stochastic meta-heuristics such as evolutionary

[1] Yield being the difference between actual and desired outcome [3]

algorithms [5].

Programmable self-assembly of chained robotic modules has previously been investigated by Griffith [6], who suggested a two-dimensional system composed of serially connected square tiles. These tiles could move around their connection-points, resulting in a flexible chain that could fold into different geometric configurations. Through the use of four tile-types with different magnetic patterns, their system could approximate any two dimensional shape. More recently, this work was extended to three dimensions, along with algorithms to approximate any three dimensional shape [7]. Inspired by this work, Knaien et al. [8] developed a system called "MilliMoteins" that consisted of a chain of electropermanent magnetic motors that could dynamically fold into different configurations. In another work [9], it was shown that serially connected soft cubes could self-assemble into a previously designed 3-D shape by being stretched by an external tension. Finally, Risi et al. [10] presented software results of a custom "printer" that folds a long ribbon of material bearing additional elements such as virtual motors and sensors. The authors evolved and optimized robots that had different morphologies and could perform different forms of locomotion.

All of these systems have demonstrated the potential of programmable self-assembly for robotics. However, the use of modules with pre-defined shapes and uniform rigidity has limited the range of morphologies that these systems could produce. In our previous work, we combined the concepts of programmable self-assembly and soft robotics [11]. We showed that by introducing components that can obtain different softness states, the diversity of achievable morphologies at a given resolution is enhanced. Also, we demonstrated that such a system could programmatically self-assemble into complex and curvilinear morphologies that other systems would require significantly more modules to produce. However, the increased design space of potential morphologies made available by this system has, as of yet, been unexplored.

In this paper, we present an automatic method that solves the problem of how to exploit this newly available design space when a target 2-D shape is given. In order to approximate any shape with high accuracy, we put emphasis on matching a 2D hamiltonian path of target shapes. As the search for a hamiltonian path through shapes is part of other research [7], we avoid this step and focus on surface shape matching here. Hence, we provide an algorithm that can approximate the surface of any 2-D shape with high accuracy using a programmatically self-assembling system composed of soft modules.

In the following, we start by briefly introducing our previous work before presenting the newly developed algorithm in detail. Subsequently, we explain the methods and experimental setup used. Finally, we characterize the performance of the implemented algorithm on randomly generated 2-D benchmark shapes.

2 Soft Cells

In our previous work [11], we introduced a model system composed of "soft cells" that can obtain different mechanical softness pre-settings. These soft cells are ini-

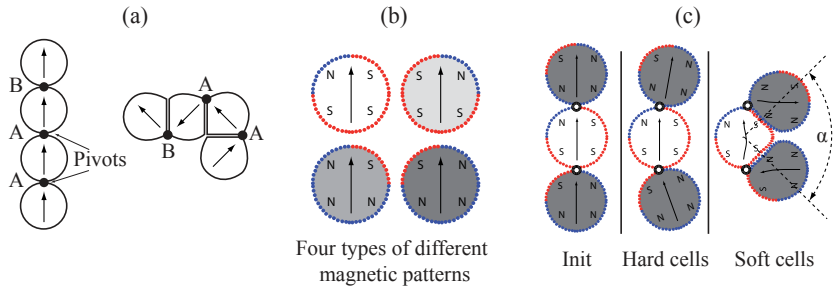

Fig. 1. (a) The basic working principle of programmable self-assembly of linear chains: modules are aligned in series and connected through permanent pivot joints. Local interactions determine the folding of the chain. (b) Design of soft cells: each cell has four magnetic sections. According to the distribution of the magnetic regions, four different cell types have been designed. (c) Schematic representation of local folding of adjacent soft cells at their initial configuration (left), at the final configuration for three hard cells (middle), and at the final configuration for three soft cells (right).

tially interconnected at permanent points and arranged as a linear chain. Every cell possesses specific characteristics in terms of softness and magnetic properties. The specific characteristics of individual cells in a sequence determines their local interactions, and ultimately defines how the self-assembly process develops.

Figure 1 displays an example of the basic self-assembly or self-folding process: cells fold about each joint once the system is released from the initial state. Joint A specifies the direction of the fold to be counterclockwise and joint B specifies the direction of the fold to be clockwise. Hence, by setting the folding sequence to be AAB the system self-assembles into the structure shown on the right of Figure 1(a).

Soft cells: For simplicity, we model a soft cell as a two dimensional object. A cell consists of a thin flexible membrane with an operating inner pressure, which defines its softness.

Local interactions: Each cell's membrane features specific connection areas that allow the cell to interact with other cells. In our system, these local interactions depend on magnetic connection areas. A minimal set of four different cell types has been designed, with four connection areas per cell type (Figure 1(b)).

Folding process Assuming a frictionless environment, self-assembly occurs as a consequence of the equilibration of magnetic interaction forces and contact mechanics between adjacent cells (Figure 1(c)).

Contact area and folding angle: Due to contact mechanics the contact area in equilibrium after folding between two highly soft cells will be larger compared to the contact area between two harder cells (Figure 1(c)). As a consequence, the angle between three adjacent cells after folding in equilibrium is also influenced by the softness of the cells (Figure 1(c)). Hence, the system's equilibrium state and final morphology is controllable through varying the softness of the cells.

3 Algorithm Description

Assumption - Sequential folding
The contact mechanics of mechanically soft, self-assembling components are typically highly non-linear [11]. In our system, composed of chains of soft cells, the non-linear effects are further compounded if the self-assembly process happens in parallel, i.e. all the local interactions between all components occur at the same time. However, in order to develop a robust algorithm that is capable of predicting the folding of a system with many components, it is essential to find a way to cope with these non-linear interactions.

One way of reducing the effects of these non-linear interactions is to force the folds to occur sequentially, e.g. by using a time-delay or a printing system such as the one presented in [6]. Because the cells then assemble one-by-one, the folds are isolated in time and cannot influence each other. This greatly limits the extent of non-linear interactions.

Modeling
Previously, we investigated and quantified the relation between cell softness, contact area and folding angle α (Figure 1(c)) of adjacent folding cells. Here, in order to predict the outcome of a single fold, we make use of these previously obtained results. We use a cubic spline interpolation to model the interplay between the contact area and the softness of a pair of folding cells.

In order to compare the algorithm to state-of-the-art systems, we also model a system comprised of cells with uniform hardness. In this system, the contact area between adjacent cells is always constant.

Algorithm Overview
The main goal of the algorithm is that for a desired resolution (i.e. number of cells or cell dimension) the cell centers match a given 2-D path as closely as possible after folding. As we assume sequential folding for the self-assembly process, we have developed an iterative search algorithm, which aims to set one soft cell after the other on the given path. In the following, we provide a more detailed description on the algorithm.

1. Read in a 2-D image. For simplicity we assume that this image contains a single shape, which can be approximated with a Hamiltonian path.
2. Process the image and extract the surface path. The surface path is extracted using common image processing techniques: canny edge detection, boundary tracing and smoothing using a 5-point moving average [16].
3. Set the softness and magnetic pattern of the first two cells: the first cell is placed at a random position on the surface path and is given a random magnetic pattern. The orientation of the first cell is set in such a way that the second cell's center is on the surface path as well. According to the distance between the two cell centers, the softness of these cells is determined. The magnetic pattern is set depending on the folding direction of the second cell.

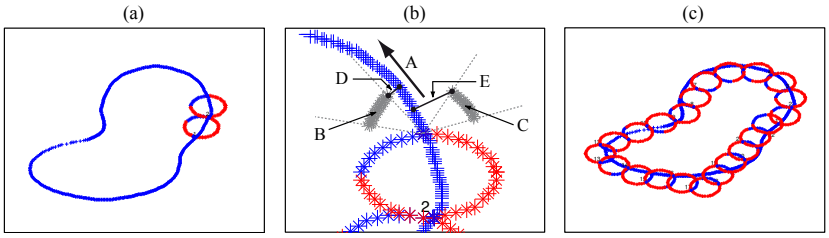

Fig. 2. Example of the algorithm intermediate steps: (a) Placement of the first two cells; (b) Finding the properties of the next cell: 'A' represents the search direction vector, 'B' depicts potential cell positions when folding counter clockwise, 'C' depicts potential cell positions when folding clockwise, 'D' = minDistCounterClockwise and 'E' = minDistClockwise. See text for details. (c) Output of the algorithm: a sequence of cells that matches the given surface.

4. Loop over the entire surface path in order to find the softness and magnetic pattern of the remaining cells. This step is further described in the next section "Algorithm details".
5. Once all of the cells have been set, i.e. their softness and magnetic patterns defined, the algorithm is finished. The result is a 1-D sequence of cells with specified softnesses and magnetic patterns.

Algorithm Details

In order to find the softness and the magnetic pattern of the next cell such that its center comes as close to the surface path as possible, the following steps are executed (see Figure 2):

1. All possible cell positions for the next cell are computed based on the softness of the previous cell. Varying the potential properties of the next cell (softness, magnetic pattern) can lead to a range of cell positions (Figure 2(b)).
2. For all potential cell positions the shortest distance to the discretized surface path is computed.
3. Two cell positions are retained: the one with the shortest distance to the surface when folding clockwise (minDistClockwise), and the one with the minimal distance to the surface when folding counterclockwise (minDistCounterClockwise)(see Figure 2(b)).
4. Check if the found cell positions are in the search direction of the surface path (see Figure 2(b)). The search direction is defined as the vector that spans from the last cell (pivot point on surface path) to the point on the surface path that is one cell diameter away. A potential cell center is assumed to be in the search direction if the angle between the surface direction vector and the vector from a cell position to the closest surface path point is smaller than 90°. If one of the two positions is not in line with the search direction, its corresponding minimal distance is set to infinite.
5. Check for the cell position with the smallest minimal distance to the surface path and return that position.

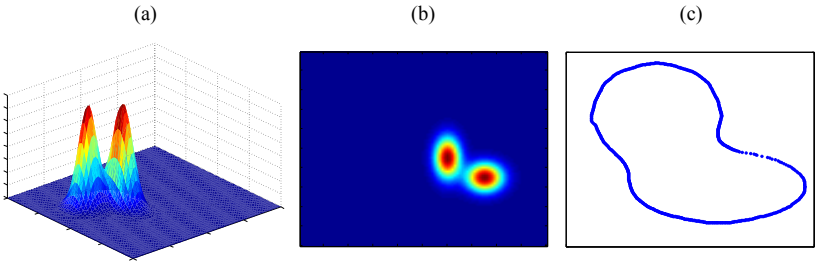

Fig. 3. Randomly generated Gaussian mixture models. Two Gaussian density points (a) shown in 3-D, (b) shown in 2-D. (c) Thresholding at an arbitrary level yields a non-uniform shape.

4 Methods

Implementation and Testing
The algorithm as described in the previous section is implemented in MATLAB® (Matlab). In order to test the algorithm, its output is tested both in Matlab as well as in our custom developed physics-based simulation tool "Soft Cell Simulator" (SCS) [12]. Using our simulation framework enables us to assess the performance in a physically more plausible and accurate way.

Experimental setup
In order to evaluate the performance of the algorithm in a fair manner, we generate randomly shaped benchmarks upon which the algorithm is tested. Random but plausible robot shapes can be obtained when using Gaussian Mixture Models [13]. In this method, a 2-D workspace contains a list of Gaussian points. Each point has an associated density and standard deviation. In order to generate a 2-D shape, the linear sum from all Gaussian points is taken and thresholded. Figure 3 illustrates an example of the method. Two Gaussian points result in a non-uniform shape. In our experiments, up to five Gaussian points are used, whereas the points may have different standard deviations. This choice of parameters typically leads to smooth, freeform shapes that potentially could be robot morphologies, as has been shown in [17].

Analysis
For assessing the correctness of the algorithm's output, we compute the geometrical accuracy between a resulting morphology and a target shape (similar to the geometric accuracy of 3-D printed models [14]):

$$\text{Geometric Accuracy} = \frac{A_{\text{target}} - A_{\text{error}}}{A_{\text{target}}} \times 100\% \tag{1}$$

where A_{target} is the surface-area of the target object and A_{error} is the surface between the centers of the folded cells and the target object. See Figure 4 for an illustration of this computation.

In order to validate the physical plausibility of our approach, we feed the algorithm's output into SCS and test the self-assembly of the cell sequence. We then determine the error between the algorithm's prediction and the physical simulation by computing the Euclidean distance between the corresponding cell centers (see Figure 7). The individual errors ϵ_i are then summed up such that:

$$\text{Accumulated Error} = \sum_{i=1}^{n} \epsilon_i \tag{2}$$

5 Results

Ten complex 2-D shape models, which were first created by Gaussian mixture models and then processed using common image processing tools, were tested to validate the effectiveness and robustness of the developed algorithm.

Geometric accuracy for randomly generated benchmarks
Figure 4 plots the geometric accuracy of the algorithm output for the randomly generated benchmarks obtained at different resolution settings (number of cells). As can be observed in the figure, the accuracy generally increases with a higher number of cells. This is because, if resolution is low, the cell centers may lay close to the desired path, but many details of the actual given shape are lost. An example of this effect is shown in Figure 5, where the approximation of the snail shape is illustrated at different resolution settings.

In order to illustrate the advantages of our system composed of cells with non-uniform softness settings, we also run the algorithm with uniform softness settings (which is the case in state-of-the-art systems [6-10]). This result is also plotted in Figure 4(b). As this plot makes clear, the geometric accuracy of the output with uniform softness settings is always lower compared to the case with non-uniform softness settings. This demonstrates the increased capabilities of the system with non-uniform softness settings to approximate shapes at lower resolution and with higher geometric accuracy.

Algorithm validation in physics-based simulation
To further validate the output of the algorithm in a physically more accurate environment, we feed the obtained cell sequences for the ten benchmarks into SCS. We then asses the error between the Matlab prediction and the physics-based simulation. Figure 6 shows an example of the difference between Matlab and SCS output. Figure 7 plots the quantified accumulated error when increasing the number of cells. As the figure makes clear, there is a linear increase of the accumulated error with each added cell. This error has two potential sources. First, because of the non-linear interactions between folding cells and the interpolation used to approximate this process in Matlab, some error occurs with each fold. Second, because of energy drift in the physics-engine (i.e. there is a gradual change in the total energy of a closed system over time, due to numerical integration artifacts that arise with the use of a finite time step [15]), there is

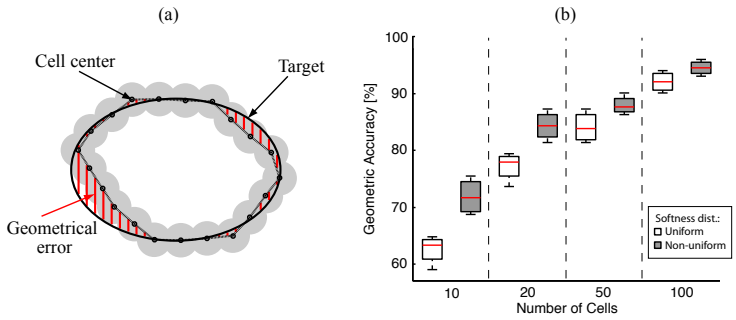

Fig. 4. (a) Calculation of geometric accuracy, (b) geometric accuracy of the algorithm's output for ten randomly generated benchmarks when allowing either a uniform or a non-uniform cell softness distribution.

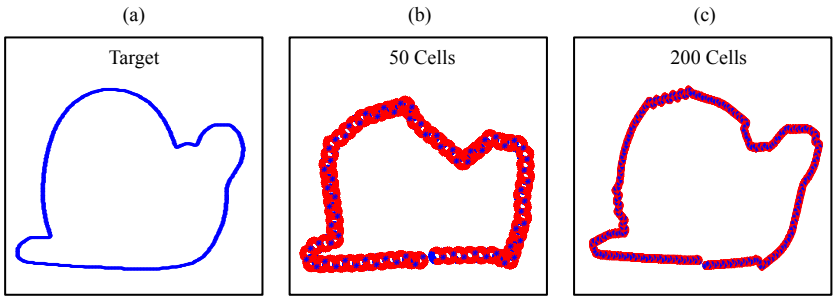

Fig. 5. Algorithm output to match the target shape of a snail. (a) Target, (b) output with resolution of 50 cells, (c) output with resolution of 200 cells.

additional error introduced at every computational step.

Finally, we compute the geometric accuracy of the result of the self-assembly process in the physics-based simulation. The result is plotted in Figure 7(c). Despite the linear increase of accumulated error between Matlab and SCS, the geometric accuracy of the physics-based simulation remains high and follows the same profile as the Matlab prediction. Thus, from this result we conclude that the algorithm, despite its simplistic modeling, is capable of predicting sequences that fold with a relatively high accuracy into desired target shapes.

6 Conclusion

In this paper, we have presented an iterative search method that finds a linear chain of soft cells that self-folds into a target 2-D shape. We evaluated the algorithm's capability of approximating diverse shapes by measuring the geometric accuracy for randomly generated benchmarks. As we have shown, the algorithm's output reaches high geometric accuracy both in Matlab and in a physics-based simulation environment.

The algorithm presented here is well-suited to be combined with a space-

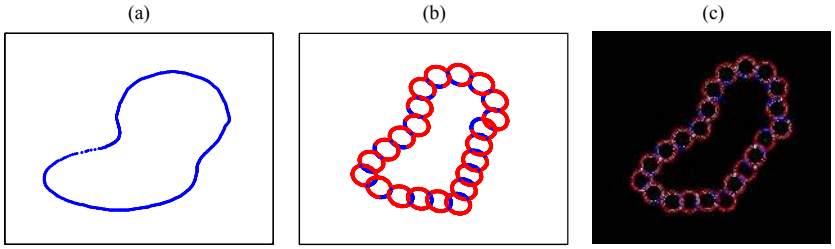

Fig. 6. Example of difference between Matlab prediction and physics based simulation environment. (a) Target, (b) Matlab prediction, (c) SCS simulation.

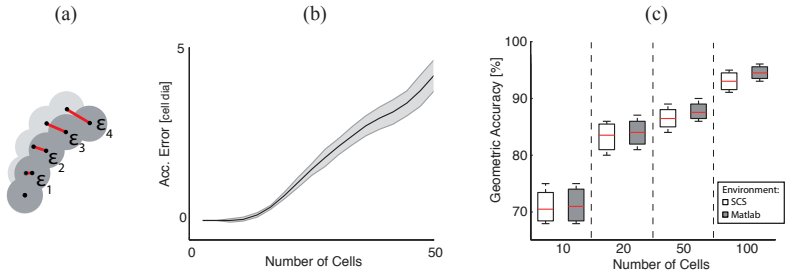

Fig. 7. (a) Calculating the error between the Matlab prediction and the SCS simulation, (b) accumulated error for an increasing number of cells, (c) geometric accuracy of benchmarks in SCS for different resolutions (number of cells).

filling algorithm such as the one presented in [7]. Then, a robot's morphology may be automatically constructed by approximating its surface using the method described here and by specifying the required filling of the body (for instance according to functional constraints such as weight, etc.). In the future, by stacking 2-D building blocks on top of each other, this principle could be further extended to the third dimension.

The results in this paper present a cornerstone in our progression towards an automatic method to design and self-assemble complex soft modular robots, because this allows for automating the step of finding a cell sequence for a target morphology while exploiting the soft characteristics of our system. As this method allows one to obtain robotic structures with high morphological complexity from a small number of components, we believe that it will be possible to create functionally complex, self-assembling robots that are well-adapted to a given task or environment.

Acknowledgements

The research leading to these results has received funding partially from the Swiss National Foundation through the National Center of Competence in Research Robotics and from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 308943.

References

1. Murata, S., Kurokawa, H.: Self-Organizing Robots. Springer Tracts in Advanced Robotics, 77 (2012)
2. Whitesides, G. M., Grzybowski, B.: Self-assembly at all scales. Science, 295, 2418-2421 (2002)
3. Pelesko, J. A.: Self assembly: the science of things that put themselves together. CRC Press (2007)
4. Gross, R., Dorigo, M.: Self-Assembly at the Macroscopic Scale. Proceedings of the IEEE, 96, 1490-1508 (2008)
5. Pfeifer, R., Lungarella, M., Iida, F.: Self-Organization, Embodiment, and Biologically Inspired Robotics. Science, 318, 1088-1093 (2007)
6. Griffith, S.: Growing Machines. PhD Thesis, MIT (2004)
7. Cheung, K. C., Demaine, E. D., Bachrach, J. R., Griffith, S.: Programmable assembly with universally foldable strings (Moteins). IEEE Transactions on Robotics, 27, 718-729 (2011)
8. Knaian, A. N., Cheung, K. C., Lobovsky, M. B., Oines, A. J., Schmidt-Neilsen, P., Gershenfeld, N. A.: The milli-motein: A self-folding chain of programmable matter with a one centimeter module pitch. IEEE International Conference on Intelligent Robots and Systems, 1447-1453 (2012)
9. Yim, S., Sitti, M.: SoftCubes: Towards a Soft Modular Matter. IEEE International Conference on Robotics and Automation, 530-536 (2013)
10. Risi, S., Cellucci, D., Lipson, H.: Ribosomal Robots: Evolved Designs Inspired by Protein Folding. GECCO '13, 263-270 (2013)
11. Germann, J., Maesani, A., Pericet-Camara, R., Floreano, D.: Soft Cells for Programmable Self-Assembly of Robotic Modules. Under review.
12. Germann, J., Maesani, A., Stöckli, Floreano, D.: Soft Cell Simulator: A tool to study Soft Multi-Cellular Robots. IEEE International Conference on Robotics and Biomimetics, 1300-1305 (2013)
13. Pernkopf, F., Bouchaffra, D.: Genetic-based EM algorithm for learning Gaussian mixture models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27, 1344-1348 (2005)
14. Jin, G.Q., Li, W.D., Gao, L.: An adaptive process planning approach of rapid prototyping and manufacturing. Robotics and Computer-Integrated Manufacturing, 29, 23-38 (2013)
15. Gans, J., Shalloway, D.: Shadow mass and the relationship between velocity and momentum in symplectic numerical integration. Physical Review E, 61, 4587-4592 (2000)
16. Gonzalez, R. C., Woods, R. E., Eddins, S. L.: Digital Image Processing Using MATLAB, New Jersey, Pearson Prentice Hall, 2004
17. Hiller, J., Lipson, H.: Automatic design and manufacture of soft robots. IEEE Transactions on Robotics, 28(2), 457-466 (2012)