Biomimetic Intelligence and Robotics 2 (2022) 100036

Contents lists available at ScienceDirect

Biomimetic Intelligence and Robotics

journal homepage: www.elsevier.com/locate/birob

Auto-generating of 2D tessellated crease patterns of 3D biomimetic spring origami structure

Yu Xing Teo [a], Catherine Jiayi Cai [a], Bok Seng Yeow [a], Zion Tsz Ho Tse [b], Hongliang Ren [c,a,*,1]

[a] Biomedical Engineering Department, National University of Singapore, Singapore, 117575, Singapore
[b] Department of Electronic Engineering, University of York, York YO10 5DD, UK
[c] Department of Electronic Engineering, Chinese University of Hong Kong, China

ARTICLE INFO

Keywords:
Biomimetic soft robotics
3D origami design
Design automation
Computer-aided design
Structural optimization
Parametric design

ABSTRACT

Computational simulations can accelerate the design and modelling of origami robots and mechanisms. This paper presents a computational method using algorithms developed in Python to generate different tessellated origami crease patterns simultaneously. This paper aims to automate this process by introducing a system that automatically generates origami crease patterns in Scalable Vector Graphics format. By introducing different parameters, variations of the same underlying tessellated crease pattern can be obtained. The user interface consists of an input file where the user can input the desired parameters, which are then processed by an algorithm written in Python to generate the respective origami 2D crease patterns. These origami crease patterns can serve as inputs to current origami design software and algorithms to generate origami design models for faster and easier visual comparison. This paper utilizes a basic biomimetic inspiration origami pattern to demonstrate the functionality by varying underlying crease pattern parameters that give rise to symmetric and asymmetric spring origami 3D structures. Furthermore, this paper conducts a qualitative analysis of the origami design outputs of an origami simulator from the input crease patterns and the respective manual folding of the origami structure.

1. Introduction

Origami techniques can create 3D structures and mechanisms from 2D planar sheets. The forms and functions of 3D origami products are determined by their underlying creases and are thus created using different folding patterns. However, given that traditional origami design involves considerable inefficient trial-and-error, the origami design of complex models can be tedious, laborious and time-consuming, especially for curved crease origami such as hyperbolic origami [1]. Hence, several computational design techniques have been developed and incorporated into origami design and modelling to develop algorithms that support the systematic and efficient designing and fabrication of origami robots and mechanisms [2,3]. In computational origami designs, the inputs to an origami simulator are an origami sheet and its crease patterns [4,5]. Different mathematical theories are used to compute and solve the origami folding patterns depending on the algorithm.

Current computational origami design software and applications can be largely categorized depending on their output. 2D generator software includes E-origami system (Eos) and Treemaker, and 3D generator software includes Origamizer, Freeform Origami and Amanda Ghassei Simulator (AGS). Eos [1,6] is a system that constructs origami on a computer and visualizes the properties of the origami. Ref. [7] shows the author introducing basic mathematical folding notions which build up the Eos model. Treemaker [8,9] is a program for designing origami bases that implements algorithms and mathematical theories to construct an origami structure with constraints set by the user [10,11]. Origamizer and Freeform Origami are systems developed by Tomohiro Tachi. Origamizer can alter the crease patterns of the origami structure while the user changes the 3D folding pattern of the origami [12–15]. Freeform Origami uses computational methods to design rigidly foldable origami [16–18]. AGS presents an explicit method for simulating a 3D origami structure from its corresponding 2D crease patterns in vector graphics format [5]. Different line opacities indicate each corresponding crease pattern for the final fold angle and different colours code for different crease types, including mountain (red), valley (blue), border (black) and cut (green).

* Corresponding author.
E-mail address: hlren@ieee.org (H. Ren).
[1] ASME Member.

https://doi.org/10.1016/j.birob.2022.100036
Received 13 October 2021; Received in revised form 28 December 2021; Accepted 14 January 2022
Available online 21 January 2022

While the aforementioned available computational origami design software and applications are useful for the modelling and visualization of origami outputs, most do not contain tools to automate the design of multiple origami systems [4,19]. The underlying crease patterns of various origami structures have specified parameters and values that can influence the folding of the origami structure [20,21]. Modifying these parameters can give rise to different structures with unique behaviours. For example, changing the parameters of the origami spring design can give rise to different forms and mechanics of the folded origami structure [22]. Hence, an external algorithm or software automating the design of multiple origami sheets and their multiple crease patterns can be useful.

The computational design for biomimetic soft robotics can solve complex problems. Recent researchers study biomimetic inspiration ideas such as worms, amoeba and octopuses to develop deployable and functional soft origami robots [23–27]. Leech has folded annular structures, which provide ductility and mechanical load-bearing ability. Recent studies demonstrated biomimetic actuators that use basic origami patterns inspired by a leech, with the structures' motion trajectory depending on different geometrical parameters [28,29]. However, recent studies in the biomimetic robotics field rarely focused on geometrical parameter analysis. Thus, this study demonstrates an automated computational method that utilizes a basic origami pattern inspired by leech to obtain various deployable, rotatable and load-bearing biomimetic soft robots after changing different geometrical parameters.

This paper introduces a computer-aided method that automatically generates multiple origami 2D crease patterns. This auto-generating system allows the user to variate different parameters, including crease opacity, angles, aspect ratios and size. Users introduce the parameters into the system through a datasheet, and a Python script extracts the input parameters from the sheet to generate the corresponding origami crease pattern, which is subsequently saved as a file in the Scalable Vector Graphics (SVG) format or equivalents. This process allows for the forms (and potential functions) of different origami structures to be compared and increases the speed of origami development [30]. Using an origami spring crease pattern [22], we demonstrate how the system works. The output of the algorithm is then used as the input to AGS [5,31]. AGS is selected due to its input/output compatibility and ease of visualization and usage compared to other existing software and applications. By giving AGS multiple inputs of different crease patterns, we can output multiple origami structures in the same file, enabling us to visualize and compare multiple these structures simultaneously.

The main contributions of this paper are as follows:

(1) Development of an automated pipeline for generating spring origami crease patterns

(2) We demonstrated our purpose based on the origami spring crease pattern by varying the parameters of the underlying crease pattern to give rise to the different origami structures and features. This was followed by a qualitative comparison between the origami design outputs of the origami simulator and the manual folding of the origami structure.

The paper is organized as follows: Section 2 introduces the auto-generating system in detail. We first describe a spring origami structure and its respective parameters of interest and introduce the user interface and python script. Section 3 compares computational folding and physical folding qualitatively and quantitatively. We also discuss the feasibility of the system and the choice of platform and strain analysis of the spring origami structure.

2. Method

The overall workflow of the auto-generating system is presented in Fig. 1. First, based on the desired origami structure, users will identify the parameters of the underlying crease patterns mandatory for the structure to exist and additional parameters of interest that they would like to vary. These parameters and specified values are then taken as inputs into the auto-generating system. The auto-generating system consists of a user interface where users can input their desired values of various parameters and a Python script that automatically generates an SVG file based on the user input. As a demonstration of how the auto-generating system can be used, this paper first focuses on generating an origami structure based on the origami spring pattern (Fig. 2(C)). Hence, the parameters used in the user interface reflect the parameters of interest identified from the origami spring crease pattern. The input parameters are customizable according to the origami crease patterns. We used a biomimetic spring origami structure in this paper as an example.

2.1. Design cylindrical and asymmetric spiralling annelids-inspired origami

This section discusses the basic origami patterns and the basic geometrical parameters of the standard annelids-inspired origami. We discuss two representatives of the basic annelids-inspired origami in this section, namely the leech-inspired origami [32] and caterpillar-inspired origami [21]. While there appear to be no well-defined origami patterns for annelids biomimetic origami, most of the studies are based on modifying the basic Yoshimura-origami patterns [21,32–43]. Studies defined the Yoshimura pattern as consisting of identical isosceles triangles arranged symmetrically in each row, producing two triangles that form a diamond [44] (Fig. 2(A)). Previous research shows that leech-inspired origami [32] (Fig. 2(B)) and caterpillar-inspired origami [21] (Fig. 2(C)) are similar to Yoshimura-origami patterns and can obtain a flat-foldable cylindrical structure [33]. Our paper introduces a cylindrical spiralling annelids-inspired actuator inspired by the leech origami pattern [32] and the caterpillar-inspired origami pattern [21,33].

Both the leech and caterpillar origami show several similarities to Yoshimura-origami. According to the leech origami study [32], the difference between their leech origami structure and the traditional Yoshimura pattern is the replacement of the isosceles triangle of the Yoshimura pattern with other scalene triangles (Fig. 2(B)), resulting in an asymmetrical Yoshimura pattern, causing directional bending motion [32]. On the other hand, the caterpillar-inspired origami pattern [21] consists of identical isosceles triangles, symmetrically connected in each row with all the triangles facing up (Fig. 2(C)) instead of forming a diamond shape. Each leg of the triangle is a valley fold, while the base and height of the triangle are mountain folds. Additionally, the caterpillar-inspired origami study shows that the locomotion mechanism of the origami resembles that of a real caterpillar [21] and can generate a flat-foldable cylindrical structure.

To produce a cylindrical asymmetric spiralling annelids-inspired origami, we combined inspirations derived from the leech origami (asymmetric) [32] and the caterpillar inspired origami (cylindrical structure) [21]. We modified the previous caterpillar-inspired origami pattern [21] to provide an asymmetric feature by replacing the "isosceles triangles" in the caterpillar-inspired origami with "multiple types of triangle", including isosceles triangles, equilateral triangles and scalene triangles. After the modification, we generated an asymmetric origami structure by using scalene triangles (Fig. 2(D)) instead of isosceles triangles. The asymmetric origami structure was able to obtained several asymmetric features (Figs. 6–8), including spiralling behaviour during the deployed stage. The folding difference between our annelids-inspired origami (Fig. 2(D)) and the caterpillar-inspired origami (Fig. 2(C)) [21] will be discussed in further detail in the results section.

After the modifications, we found that our origami structure appears to contain similar geometrical parameters used in common annelids-inspired origami, specifically the height to base ratio and apex angle of the triangle [21,32–43]. For example, in the leech origami study [32], the major geometrical parameters are the apex angles of the triangle (denote as β_1 and β_2) between the mountain fold crease and the valley fold crease of the unit cell [32] (Fig. 2(B)). By controlling the apex

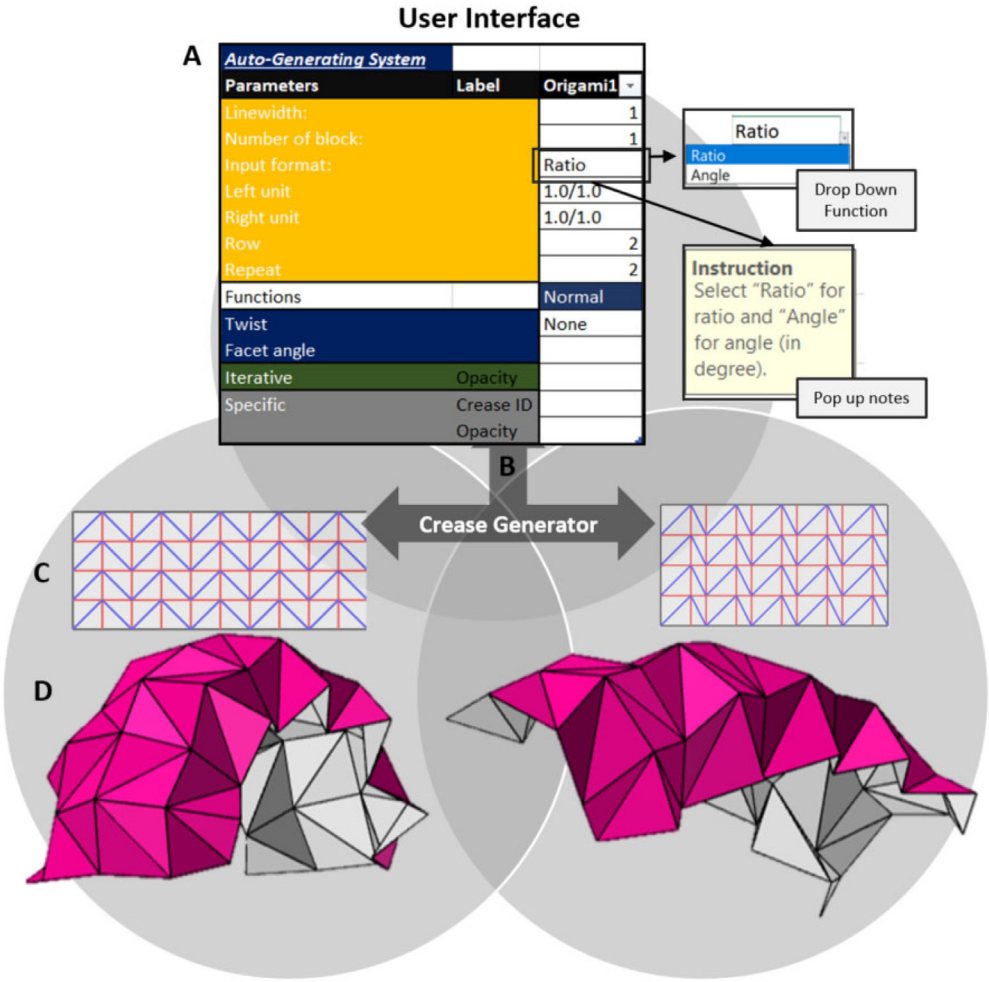

Fig. 1. Overall workflow of the auto-generating system. (A) The user first inputs the values of the parameters into the Excel user interface. The dropdown function is used in some parameters to standardize the user inputs. Detailed instructions are provided using Excel pop up notes. (B) The Python script extracts the user inputs and constructs the 2D origami crease patterns. (C) The origami crease patterns are stored in an output file (in SVG format). (D) The output file can be used as an input for AGS to visualize the folded origami structure based on the crease pattern.

angles, β_1 and β_2 (Fig. 2(B)), the leech origami structure can produce directional bending motion [32].

On the other hand, the study on the caterpillar-inspired origami structure [21] claimed that the height to base ratio is the major geometric parameter of their origami pattern, denoted as a/b ratio (Fig. 2(C)) [21]. Parameter a/b is defined as the ratio of the height (denoted as a) to half of the base length (denoted as b) of the triangle. In this paper, to replace the "isosceles triangles" in the caterpillar-inspired origami [21] with "scalene triangle", the major geometric parameter a/b ratio needs to be replaced with the H/R and H/L ratios (mentioned in the following section), or interpreted as α and β angles (Fig. 2(D)). After this modification, our asymmetric spring origami pattern will produce spiralling behaviour, which we will discuss in the next section.

2.2. Spring origami pattern and geometrical parameters

Our origami spring crease pattern consists of repeated units defined as a rectangular block divided into two parts (left and right), as shown in Fig. 3(A). Each part would comprise of two scalene triangles after replacing the isosceles triangles with scalene triangles.

In this paper, the following four parameters of interest in each unit were identified: (i) spring origami angle ratio, (ii) horizontal length ratio, (iii) number of repeats and (iv) number of rows. Both the spring origami angle ratio and horizontal length ratio are parameters that determine whether the final folded origami structure will be symmetric or asymmetric. The row parameter yields a closed or open origami square structure based on the given origami angle ratio (Fig. 3(B)–(C)). Modifying the repeat parameter can yield different lengths of the final origami spring structure. Modifying the four identified parameters of the crease patterns can give rise to different spring origami structures with potentially unique behaviours and properties. The following section will explain each parameter included in the spring origami structure generator.

Spring origami ratio and angle

As mentioned previously, each unit of the crease pattern can be divided into two parts (left and right). Each part can be characterized by its half-angle (alpha for left, beta for right) or the ratio of its height (H) to its horizontal base length (L for left, R for right). By using different ratios and angles, different spring origami structures can be generated. In symmetrical spring origami crease patterns, the left and right parts of each unit are identical. Fig. 4(A) shows an example of a symmetric spring origami crease pattern where $H/L = H/R = 3/4$ on both sides of the unit. Asymmetric spring origami crease patterns consist of unequal ratios or angles within each unit. Fig. 4(B) shows an example of an asymmetric spring origami crease pattern where $H/L = 4/3$ and $H/R = 3/4$. Unlike symmetric crease patterns that give rise to straight and linear origami spring structures (Fig. 4(A)), asymmetric crease patterns can give rise to twisted origami structures (Fig. 4(B)). A larger difference between H/L and H/R will result in more intense twisting. Thus, in this paper, $\frac{H/R}{H/L}$ plays an important role in the origami

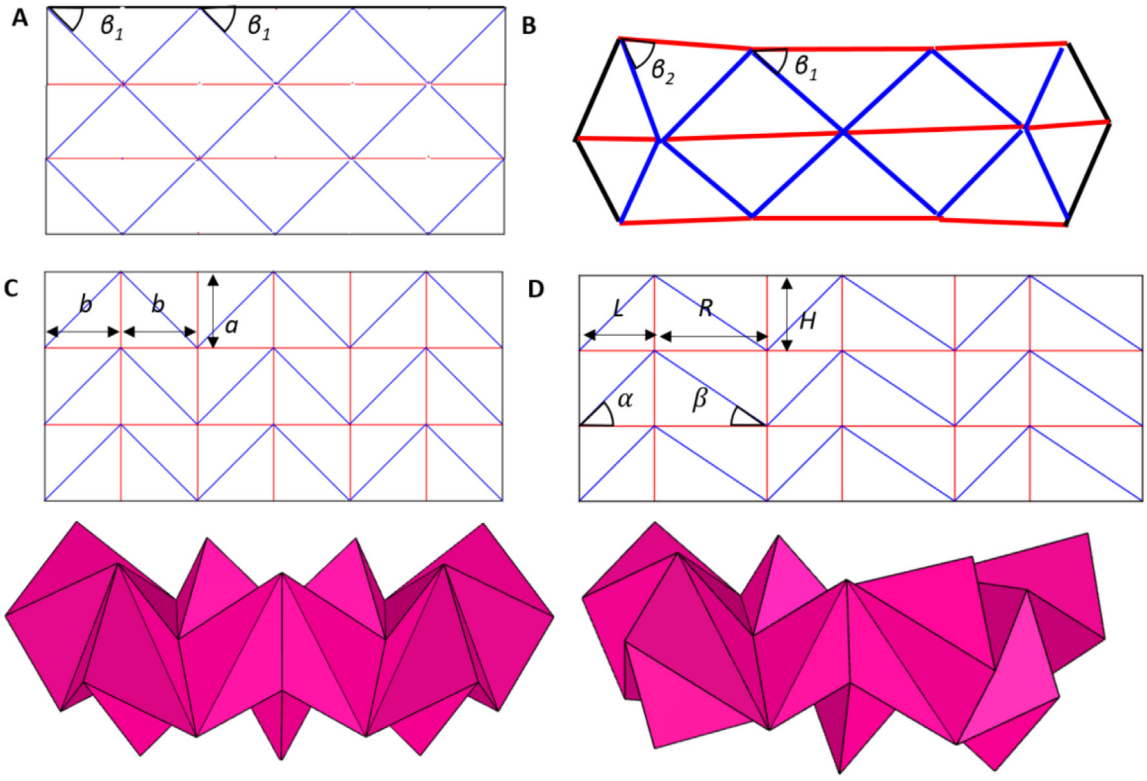

Fig. 2. Basic origami patterns and common geometrical parameters in annelids-inspired origami studies. Red lines indicate mountain folds and blue lines indicate valley folds. (A) Common Yoshimura-origami pattern consists of identical isosceles triangles, symmetrically connected in each row so that two triangles form a diamond shape. (B) Origami pattern and geometrical parameters β_1 and β_2 of the leech origami. (C) The crease pattern of caterpillar-inspired origami. It shows the geometrical parameter a/b of the caterpillar-inspired origami structure. (D) The origami patterns and geometrical parameters of our annelids-inspired spring origami.

twisting feature, which we will show in the following section. Since H is identical in each origami structure, $\frac{H/R}{H/L}$ can be expressed as L/R. More details of the twisting feature will be discussed in the results (Section 3(A)).

Spring origami repeats and rows

The spring origami crease pattern comprises repeated spring origami units in rows (M_r) and columns/repeats (N_r). Rows are defined as the number of units in each repeat. Depending on the angle/ratio of each unit, the number of rows will determine if the consequent folded structure is closed or open. For example, the crease pattern in Fig. 3(B) consists of four rows with a symmetrical angle (or height to base ratio) to yield a closed square structure. Repeats are defined as the number of units in each row and will determine the final length of the structure. Similar to how the elasticity of a physical spring increases with the number of coils, the elasticity of the spring origami also increases with the number of repeats. This will intensify several properties, such as stretchability and compressibility. The dimension of the spring origami crease pattern can be short-handed to $M_r \times N_r$. Fig. 3(B) shows an origami spring crease pattern with 4×5 origami spring units, consisting of four rows and five repeats.

2.3. Overview of the auto-generating system

User interface

As a preliminary demonstration, the user interface consists of an Excel sheet (Fig. 1(A)) where users can choose to input their parameters of interest in the first column and their desired values or features in the following columns depending on how many structures they would like to generate concurrently in a single SVG file (Fig. 1(C)). To provide a more user-friendly experience, we used the drop-down list option (Fig. 1(A)) to provide a more standardized input. The user interface also provides detailed step-by-step pop out instructions (Fig. 1(A)) when the user intends to input the values. In addition, we used different colours to label those parameters, where the details of each parameter is described in the Excel user interface template (Fig. 1(A)). As shown in the first column of Fig. 1(A), two categories of parameters are identified for the origami spring crease pattern. Parameters labelled in yellow (line width, number of blocks, input type, left unit ratio/angle, right unit ratio/angle, rows and repeats) are mandatory parameters to generate the structure. The parameters labelled in other colours are optional parameters and features that may result in different folding mechanisms that yield structures with various characteristics and functionalities. Most of the mandatory parameters have already been discussed in the previous sub-section.

Based on the inputs and parameters specified by the user, a Python script is written and executed to automatically and concurrently generate the different structures in a single SVG file (Fig. 1(B)). This file is saved in the file name specified by the user. Users can find the generated file through the correct file pathway and use the SVG file to interact with the AGS (Fig. 1(C)) or print it on paper.

Python script

The script preset constraints to fulfil the spring origami units and crease patterns (Fig. 5). In the first part (Fig. 5(A)), the Python script extracts the input parameters from the Excel sheet and uses the parameters to modify the spring origami unit and/or the crease pattern under the preset constraints. The parameters include the ratio (angle), rows, repeats, twist, facet angle and crease opacity level. The Python script also takes a second Excel sheet consisting of hexadecimal colour code data, including the opacity level (in percentage) with the respective hexadecimal colour code of opacity as an input. The script extracts the desired parameter value from the first Excel sheet and uses the second Excel sheet to convert hexadecimal colour code and opacity level in a percentage format.

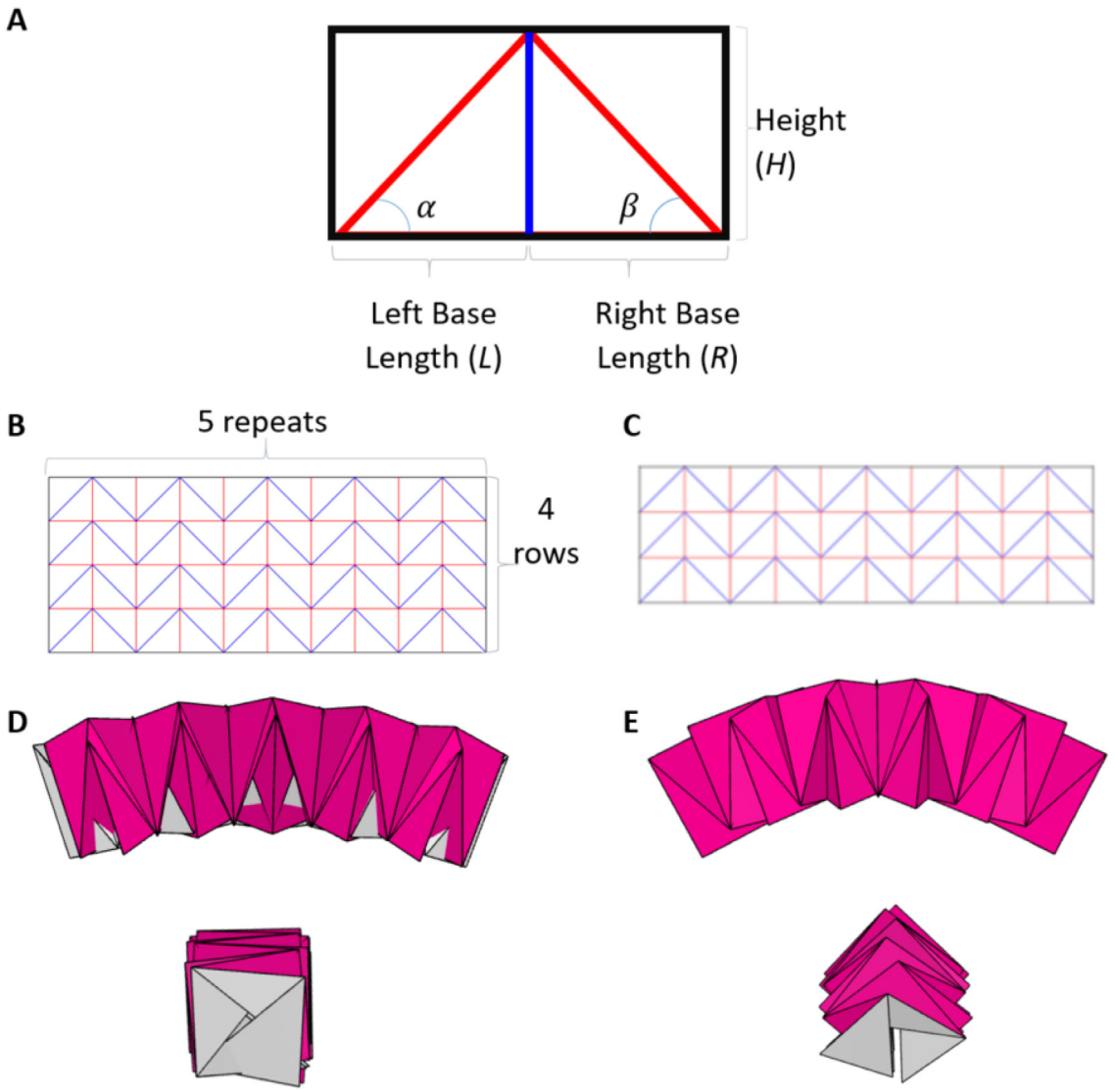

Fig. 3. Parameters of spring origami structures. (A) Single origami spring unit. The origami spring angle ratio is determined as the ratio of α to β, and the horizontal length ratio is determined as the ratio of the left base length (L) to the right base length (R). (B) Example of a standard closed surface origami structure consisting of four rows and five repeat units with a symmetrical angle to yield a closed square structure (C) Example of standard origami structure consisting of three rows and five repeat units with a symmetrical angle to yield an open square structure. (D) and (E) The front view (top) and the side view (bottom) of the AGS folded origami structures which correspond to the origami structures shown in (B) and (C) respectively.

In the second part (Fig. 5(B)), the script uses one of three different functions, namely Normal function, Specific function and Iterative function. In this paper, we will use the Normal function. Other functions are not the focus in this paper.

The third part of the script (Fig. 5(C)) uses the parameters retrieved in the first part to generate the respective spring origami crease patterns. In this part, there are three modules (*M1*, *M2* and *M3*). The first module (*M1*) generates the left part of the spring origami unit, and the second module (*M2*) generates the right part of the spring origami unit. The third module (*M3*) generates the spring origami crease pattern by repeatedly calling *M1* and *M2*. The final output file is generated in SVG format. Alternatively, the file can be converted into a different format to interact with other online computational folding platforms, like Freeform Origami or Origamizer.

Apart from the major parameters we introduced, our Auto-Generating System can also provide additional functions that allow users to modify specific origami creases with user desired crease opacity. Crease opacity is a percentage in the range of [0,100]. An origami crease with 0% crease opacity represents a transparent crease generated by our system. In contrast, a crease with 100% crease opacity indicates an opaque crease generated by the system. In AGS, creases with different opacity indicate different maximum fold percentages [5]. For instance, the higher crease opacity will have a higher folding percentage in the AGS folding (Supplementary S1).

In our system, the user can modify the crease opacity in two different manners: (i) Specific function and (ii) Iterating function.

The Specific function allows the user to input the specific "seat number" of the crease in the origami structure that needs to be modified, followed by the specific crease opacity (Supplementary S1). Users can input multiple pairs of "seat number" and crease opacity for one spring origami structure. For instance, the user may input the 1st and 2nd crease, with crease opacity 5% and 40%. The system will generate a normal spring origami with 5% crease opacity for the 1st crease and 40% crease opacity for the 2nd crease (Supplementary S1). In addition, the system also allows users to generate multiple modified origami at the same time (Supplementary S1).

On the other hand, the Iterative function creates N derivative structures from a normal spring origami with N creases. The iteration function can generate structures that can each have different modified

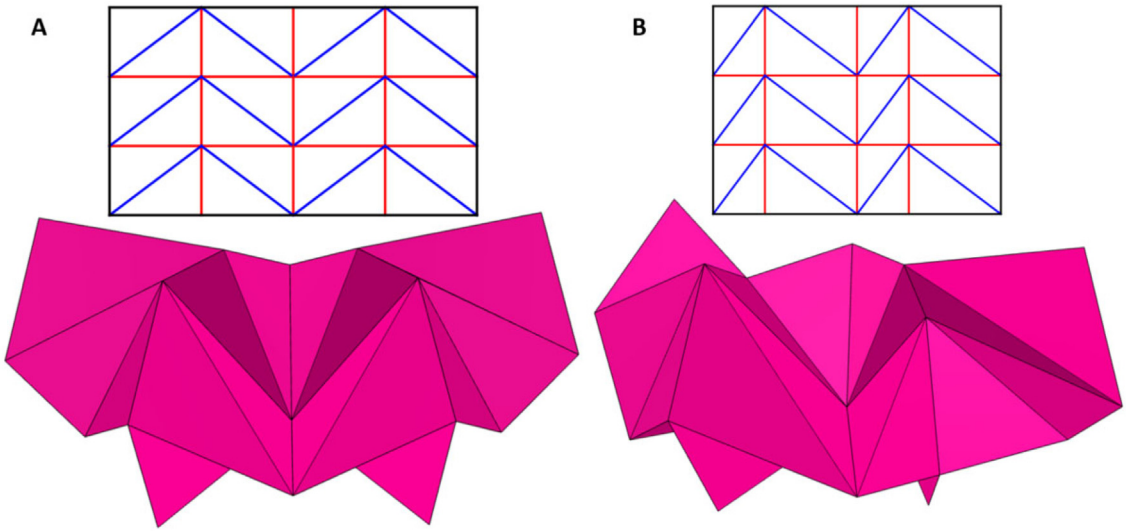

Fig. 4. Comparison of symmetric and asymmetric folding of spring origami. (A) 2D symmetric origami crease pattern where $H/L = H/R = 3/4$ and $L/R = 1$ followed by the corresponding folded origami structure in AGS. (B) A 2D asymmetric spring origami structure with left (L) ratio 4/3, right (R) ratio 3/4 and $L/R = 9/16$ followed by the corresponding folded origami structure in AGS.

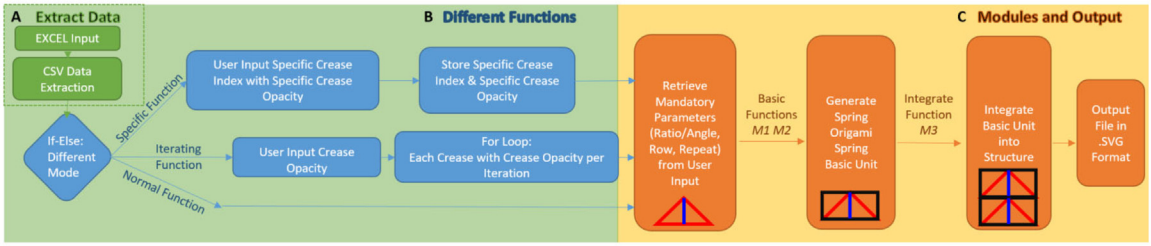

Fig. 5. Flowchart of the Python script algorithm. Python script is introduced in three parts: extract data, different functions, similar modules and output. (A) The user inputs desired values of the parameters into the Excel sheet (in CSV format), and the Python script extracts the desired values of the parameters from the CSV file. The Python script uses the user input to construct the origami structure. (B) The Python script has three functions, namely Normal function, Specific function and Iterative function. (C) Different functions share similar modules to construct the basic spring origami units and integrate the basic units into desired origami structures. Multiple desired origami structures are stored in an output file (in SVG format).

creases. For instance, the user may choose a 0% crease opacity to generate an origami structure consisting of three rows and two repeat units (36 creases) using the iterating function. The system will then generate 36 derivative structures in a single output file, with each structure having one crease with 0% crease opacity. None of the 36 derivative structures generated by the system will have the same transparent crease (Supplementary S2). The single output file can be subjected to AGS, allowing all the derivative structures to be folded simultaneously (Supplementary S2). This function enables simultaneous comparison of the folded derivative origami structure with the different modified creases. While the Specific function requires the user to specify the crease that needs to be modified, the Iteration function will automatically generate multiple origami structures with all possible creases that can be modified.

This paper focuses on the major parameters and the Normal function. The details of other additional functions can be found in our GitHub (Repository name: Auto-Generating System-Biomimetic-Spring-Origami). The Excel user interface template and our Auto-Generating System Python Script (published as a python library) are also accessible on GitHub under MIT license. More details and user guidance for the Excel interface can be found in the user interface template in GitHub.

3. Results and analysis

The auto-generating system developed in this paper can be beneficial in the design process of origami crease patterns. To demonstrate the benefits of computational origami simulation in the design and prototyping of multiple origami structures, the features observed from the simulation of the folded origami crease structures generated by the AGS using the SVG files developed by the proposed system as input (computation outputs) are compared with the features observed from the physical folding of the structures. The findings are then classified based on the generated origami spring structures into three categories: (i) qualitative analysis of the system (twisting), (ii) quantitative analysis of the system (closing angle), (iii) feasibility of the system and the choice of the platform and (iv) the strain analysis of spring origami structure.

3.1. Twisting — Qualitative analysis of the system

Cylindrical twisting is an inherent property of every asymmetric spring origami and generates block rotation of the structure (Fig. 6). As mentioned earlier, the direction and magnitude of twisting are proportional to the difference between the left ratio (H/L) and the right ratio (H/R).

- When $H/L < H/R$, i.e., $\frac{H/R}{H/L} = L/R > 1$, the structure features a counter-clockwise (CCW) twist, where the magnitude of twist is proportional to $\frac{H/R}{H/L}$.
- When $H/L > H/R$, i.e., $\frac{H/R}{H/L} = L/R < 1$, the structure features a clockwise (CW) twist, where the magnitude of twist is proportional to $\frac{H/R}{H/L}$.

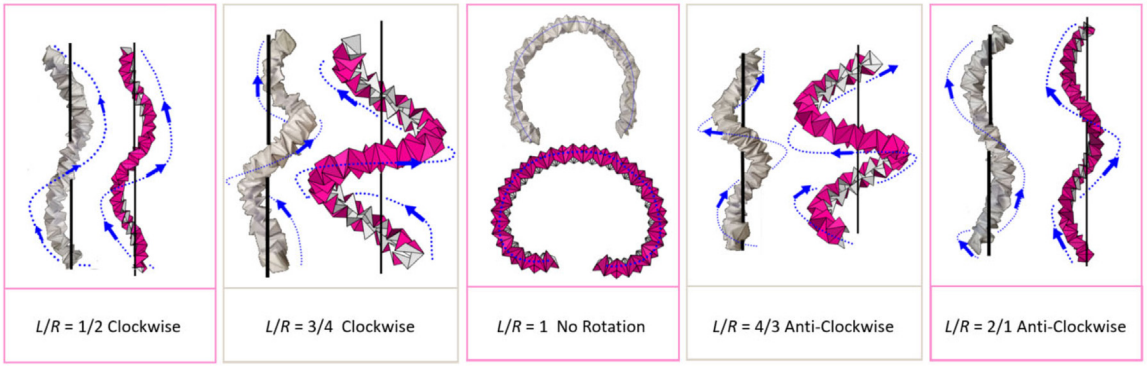

Fig. 6. Comparison of physical folding with computational folding in terms of twisting. Spring origami structure with L/R less than, equal to or greater than one. When L/R is greater than one, it will have a counter-clockwise block rotation. When L/R is less than one, it will have a clockwise block rotation. When L/R is equal to one, no rotation occurs and the spring origami structure shows curvature due to surface tension.

- When $H/L = H/R$, i.e., $\frac{H/R}{H/L} = L/R = 1$, the structure features no twist (symmetric structure), so the magnitude of twist is zero.

From Fig. 6, the twisting of the asymmetrical origami spring structure can be observed in both computational output and physical folding.

3.2. Quantitative comparison between physical folding and AGS folding

To demonstrate the similarity between AGS folding and physical folding in different perspectives, we conducted four independent experiments which would allow us to compare the two types of foldings. In this paper, our focus is to analyse the asymmetric behaviour of our spring origami, more specifically, the spiralling feature of our annelids-inspired actuators. The major parameter, the L/R ratio, which causes the asymmetric features, was studied in this validation section. The experiments are classified as single base folding validation and structure folding validation. Single base folding validation includes validation of the single base folding and closing angle of the single base. The normalized radius of the twisted spring origami and the azimuthal angle of the asymmetric twisted origami are classified as folding structure validation.

In this validation, our focus is on comparing the behaviour of single base physical folding and AGS folding within different fold percentages. Single bases with different L/R ratios were used in this validation to demonstrate the L/R ratio as one of the variables in single base folding. The position difference between the two hinges during different fold percentages was measured and it is defined as

$$\frac{Y_{OA}}{Y_{OB}} \tag{1}$$

$$Y_{OA} = y_A - y_O \tag{2}$$

$$Y_{OB} = y_B - y_O \tag{3}$$

where Y_{OA} denotes the difference of the y-axis coordinate between the origin (point O in Fig. 7(A)) and the right hinge (point A in Fig. 7(A)), and Y_{OB} denotes the difference of the y-axis coordinate between the origin and the left hinge (point B in Fig. 7(A)). The y-axis coordinate of point A, point B and point origin are denoted as y_A, y_B, y_O respectively.

The experiments are run in triplicates and the coordinates of each fold percentage were measured by Tracker (open source software) [45]. The validation results show AGS and physical folding are similar among different fold percentages and different L/R ratios (Fig. 7(A)). The $\frac{Y_{OA}}{Y_{OB}}$ values of physical folding are ±2.12% (in average) compared to the $\frac{Y_{OA}}{Y_{OB}}$ values in AGS folding among different L/R ratios (Table 1). In addition, less asymmetric origami and symmetric origami tend to have $\frac{Y_{OA}}{Y_{OB}}$ ratios closer and equal to one, respectively. This hence concludes that the L/R ratio is one of the major variables in single base folding.

Using the asymmetric spring origami structure as the base, hook structures capable of gripping via compressing and extending the spring structure can be formed. At rest, the spring origami structure behaves like a spring, folding into a hook-like structure with the self-locking property upon compression. This section of the paper focuses on illustrating one of the asymmetric hook structure's features, the closing angle. The closing angle in this experiment is defined as the angle between the horizontal edge and the free flap edge (Fig. 7(B)). The closing angle is related to the horizontal length ratio of the single spring origami unit. Experiments are conducted in triplicates, and the results are shown in Fig. 7(B).

The simulation conducted under the same conditions (triplicates with identical mandatory parameters) as in the experiment is in agreement with the experimental results (Fig. 7(B)), validating the use of model-based simulation to guide the design of tessellated origami crease patterns with the proposed auto-generating system. Symmetrical behaviour is observed in the AGS folding. However, it is not observable in physical folding because, while varying the ratio, the physical folding data points have great variation relative to the second-order trend line ($R^2 = 0.9543$) compared to the AGS folding data points ($R^2 = 0.9161$). The closing angle in physical folding is ±4.00% (in average) compared with the closing angle in AGS folding among different L/R.

In structure folding validation, we focus on investigating the spiralling properties of our annelids-inspired origami. Firstly, we compared the normalized radius of the physical and AGS folded origami followed by the azimuthal angle of the asymmetric physical and AGS folded origami.

The normalized radius of the spiralling origami is calculated by

$$R_r = \frac{r_1}{(r_2 - r_1)} \tag{4}$$

where normalize radius is denoted as R_r, the inner and outer radius of the spiralling origami (Fig. 8(C)) are denoted as r_1 and r_2 respectively. The intuition behind the normalized radius can be exemplified in the difference in how a snake of a given diameter would coil differently around a thick tree trunk as compared to a thin tree branch.

The radius of each spiralling origami was measured (triplicates) using ImageJ [46]. According to the validation results in Fig. 8(C), annelids-inspired origami structures show that both physical and AGS folding have a similar trend among different L/R ratios. Statistical analysis shows a ±2.50% (in average) difference of the R_r values between physical folding and AGS simulated folding among different L/R ratios (Table 1). Additionally, an increase in the R_r value is seen when the L/R value is closer to one, with the spring origami experiencing a greater R_r value increment when the L/R ratio is closer to one (Fig. 8(C)).

Azimuthal angle is a basic geometrical parameter to analyse the helix properties (such as twining angle) of twisted origami [47]. The azimuthal angle (ϑ) is defined as the direction change (i.e. the angle

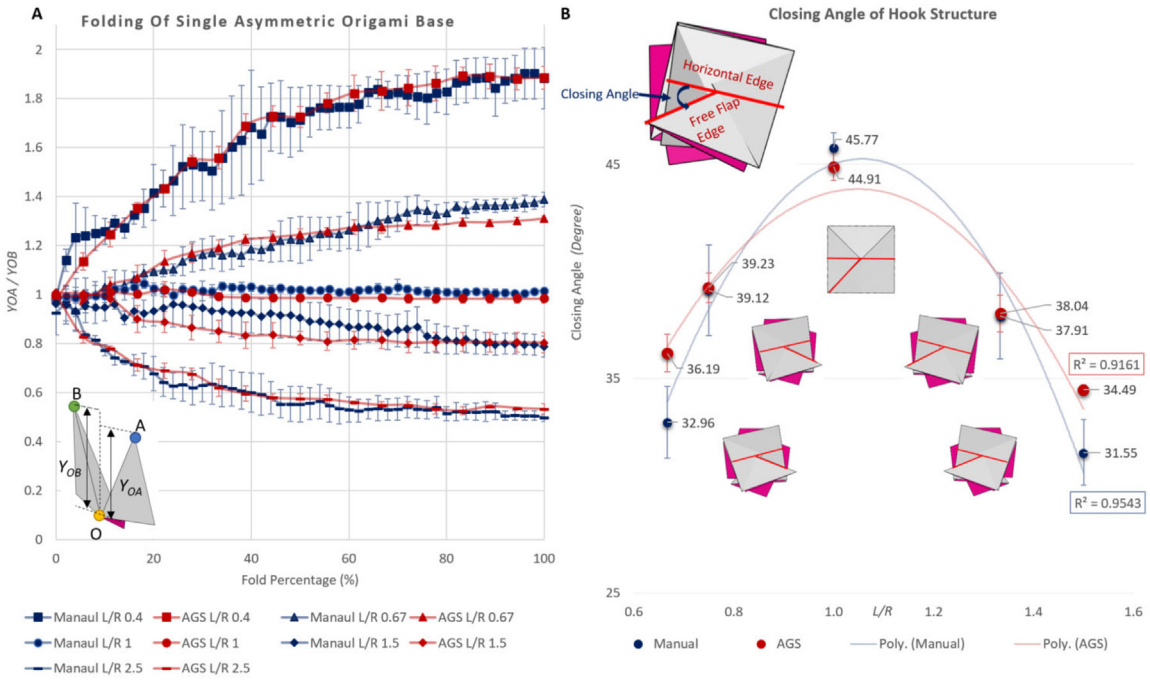

Fig. 7. Physical folding shows similar folding compared with AGS folding in certain perspectives. (A) Comparison between AGS and physical folding in terms of the folding of single spring origami base. We conducted the folding for a single origami base with different L/R ratios. The differences of the Y coordination between point O with A and point O with B are denoted as Y_{OA} and Y_{OB} respectively. The Y_{OA}/Y_{OB} ratios at different fold percentages are calculated and the results are compared between AGS and physical folding. Results show AGS and physical folding are similar among different fold percentages and different L/R ratios. In addition, less asymmetric origami and symmetric origami tend to have Y_{OA}/Y_{OB} ratios closer and equal to one, respectively. (B) Comparison of physical folding with computational folding in terms of closing angle. Spring origami structures with different L/R values will have different closing angles. The closing angle is defined as the acute angle between the free flap edge and the horizontal edge. The closing angles of both physical folding and computational folding were measured. Second-order polynomial fitting was used for both foldings.

between two vectors) between the two vectors OP0' and OP1' on the x–y plane. The vector, OP0', is the vector connecting the origin (point O in Fig. 8(D)) with the orthogonal projection (point P0' in Fig. 8(D)) of the top end of the spiralling spring origami (point P0 in Fig. 8(D)) [47]. The vector, P1'O, is the vector connecting the origin with the orthogonal projection (point P1' in Fig. 8(D)) of the bottom end of the spiralling spring origami (point P1 in Fig. 8(D)). The azimuthal angle is the total direction change between two vectors before and after the actuation of the asymmetric origami that can be larger than 360° as indicated in blue in Fig. 8(D).

We observed that the azimuthal angle of our annelid-inspired origami is dependent on the number of repeats in each row of the spring origami (denoted as N_r in the previous section). Spring origami structures with the same L/R ratio and different N_r were folded in both physical and AGS methods. The azimuthal angle of each folded origami structure was measured (triplicates) using ImageJ [46] and the results showed that both folding methods have similar azimuthal angle among different N_r (Fig. 8(F)). The azimuthal angle in physical folding is ±2.73% (in average) compared with the azimuthal angle in AGS folding among different N_r (Table 1). Furthermore, the results demonstrated a positive correlation between the azimuthal angle and N_r.

From these quantitative analyses, we observed a similarity between the behaviour of the system-generated origami and the origami generated via physical folding in different perspectives, including single-base folding (Fig. 7(A)), closing angle (Fig. 7(B)), the normalized radius of the spiralling origami (Fig. 8(C)) and azimuthal angle of the asymmetric origami (Fig. 8(F)). In addition, each quantitative analysis consistently showed less than ±4% percentage difference between physical and AGS folding (Table 1). This proves that the proposed system (the combination of our Auto-generating system and AGS) can generate similar folding behaviour in certain perspectives.

Table 1 summarizes that the average percentage difference of each quantitative analysis is equal or smaller than 4%. The percentage difference is defined as the percentage of the absolute difference between physical folding value and AGS folding value which is normalized by AGS folding value.

Table 1
The percentage difference between physical and AGS folding among different quantitative analyses.

Experiment	Single base folding	Closing angle	Normalize radius of spiral (R_r)	Azimuthal angle (ϑ)
Average percentage diff	±2.12%	±4.00%	±2.50%	±2.73%

3.3. Using auto-generating system to design and optimize biomimetic spring origami structure

Previous research designed a worm-like pneumatic spiral soft robot which is able to operate in a confined environment [47]. This section aims to design a spiral soft robot with similar design requirements provided in [47] (with some modifications). The task of such soft spiral robots would be to wrap about a given object within a predefined spatial constraint, and thus, we define the origami design requirements to be (i) 540° azimuthal angle, (ii) an ability to spiral around a cylindrical tube with a 1.9 cm radius r_1 and (iii) increasing the overall radius r_2 to 4.3 cm (Fig. 8(A)). The width of the spring origami is thus 2.4 cm (equal to the space between the outer radius (r_2) and inner radius (r_1), i.e. $r_2 - r_1$). Since the inner radius r_1 is 1.9 cm and the outer radius r_2 is 4.3 cm, according to Eq. (4), the targeted R_r for the origami is 0.792. Subsequently, we can identify the geometrical parameters that achieve the targeted azimuthal angle and the targeted R_r. Referencing the experiments in the validation section, we found that the azimuthal angle depends on the number of repeats (N_r) while the R_r is dependent on the L/R ratio. N_r and L/R are geometric parameters that can be optimized. This approach to geometrical parameters can be extended beyond our origami pattern to other variations of origami worm structures

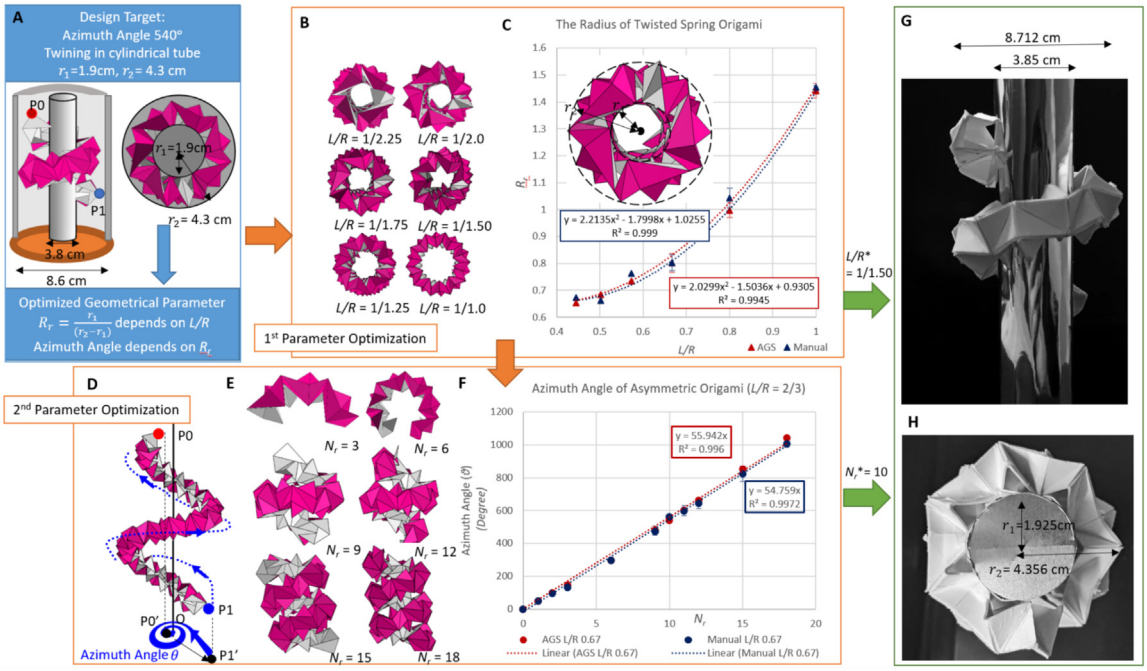

Fig. 8. Design Optimization using Auto-generating System. There are three steps in the design optimization experiments, namely identifying geometrical parameters which need to be optimized (in blue), parameters optimization (in orange) and reconstructing the biomimetic origami with optimized parameter values (in green). (A) The target origami design requirements including azimuthal angle 540° (from P0 to P1) and the origami twines in a cylindrical tube which has an inner radius of 1.9 cm and outer radius of 4.3 cm (These requirements were used in the designing process of Mei Yang et al. soft robotics study). Geometrical parameters which correspond to those requirements were identified. For instance, the L/R ratio and number of repeats (N_r) are the parameters for generating spring origami with different twisting radius and different azimuthal angles. For multiple parameters optimization, one parameter is chosen first for optimization before moving on to the others. Since the distance between the inner and outer ratio is 2.4 cm, we fixed the width of the target origami at 2.4 cm, giving a target R_r of 0.792 (calculated using Eq. (4)). (B) The normalized radius of twisted spring origami is first optimized by generating spring origami within L/R 1/2.25 to 1/1.0. Results show that the spring origami with L/R equating to 2/3 has the closest R_r ratio to the target value (0.792). (C) AGS-folding and manual-folding with certain L/R ratios were conducted in triplicates. It shows that the physical folding results are similar to AGS folding. Optimized L/R was identified and used in azimuthal angle optimization. (D) The illustration of azimuthal angle which is the total direction change of spring origami before and after actuation, i.e. the angle difference between vector OP0' and OP1', the azimuthal angle in subfigure D is larger than 360°, indicated as ϑ. (E) Optimization of azimuthal angle was conducted using the spring origami with optimized L/R ratio and width. The azimuthal angle was optimized by varying the number of repeats (from 3 to 18 with an increment of 3). (F) We selected and manually folded some candidates of the optimization experiment and found they have similar azimuthal angles compared with AGS folding. The results show that the asymmetric spring origami with 10 bases has the closest azimuthal angle to 540°. (G-H) The front view and the top view of the manually folded asymmetric spring origami with the optimized parameter values L/R and N_r. Both G and H show that the optimized spring origami meets the design requirements shown in subfigure A.

or design requirements. To summarize, parameter optimizations are conducted to find the optimal geometrical parameters (N_r and L/R) to generate an origami structure that meets the design requirements, i.e. spring origami structure with targeted R_r and azimuthal angle.

In this study, we chose a simplified Coordinate Descent with the Gauss–Seidel model [48,49] approach to enable the workflow but finding the best optimization approach is not the focus of this work. The optimization method introduces in this work employs the following equations (Eqs. (5)–(10)).

$$R_r = a(L/R) \tag{5}$$

where function a takes L/R as input variable and returns the R_r value. The L/R is the independent variable for R_r. The definition of R_r is shown in Eq. (4).

$$g(L/R) = \left| \frac{a(L/R) - \widetilde{R_r}}{\widetilde{R_r}} \right| \tag{6}$$

The target value of R_r (0.792 in our optimization example) is denoted as $\widetilde{R_r}$. The function g takes in a specific L/R value and calculates the normalized absolute value between the resulting R_r (with specific L/R) and targeted R_r ($\widetilde{R_r}$). Function g is a cost function that maps alternative L/R values and returns some "cost" (normalized absolute value) associated with the L/R values.

$$L/R^* = \arg\min_{L/R} g(L/R) \tag{7}$$

The *argmin* function assigns alternative L/R values (with uniform step size) into function g. The *argmin* function returns the specific L/R value, resulting in the smallest "cost" (i.e. minimizes the normalized absolute value calculated by function g) compared with other alternative L/R values. The specific L/R value is called as the optimized value of L/R (denoted as L/R^*).

After optimizing the L/R value, azimuthal angle (ϑ) is optimized by following equations (Eqs. (8)–(10)). Similar to how Eqs. (5)–(7) optimized the values for L/R to L/R^*, Eqs. (8)–(10) performs the optimization for N_r.

$$\vartheta = b\left(N_r | L/R^*\right) \tag{8}$$

Both N_r and L/R are the independent variables for ϑ. Since L/R is optimized. The function b takes in N_r as the input variable and returns the ϑ value, where the L/R variable has been previously optimized thus ϑ is dependent only on N_r.

$$f\left(N_r | L/R^*\right) = \left| \frac{b\left(N_r | L/R^*\right) - \tilde{\vartheta}}{\tilde{\vartheta}} \right| \tag{9}$$

The target value of ϑ (540° in our optimization example) is denoted as $\tilde{\vartheta}$. Given the optimized L/R (L/R^*), the function f takes in a specific N_r value and calculates the normalized absolute value between the resulting ϑ (with specific N_r) and targeted ϑ ($\tilde{\vartheta}$).

$$N_r^* = \arg\min_{N_r} f(N_r | L/R^*) \tag{10}$$

Given the optimized L/R (L/R^*), the *argmin* function takes in the function f and returns the optimized N_r value, which minimizes the function f (i.e. minimizes the normalized absolute value). The optimized value of N_r is denoted as N_r^*.

In this method, the L/R of the geometrical parameters was chosen to be optimized first (Eqs. (5)–(7)). Once the L/R parameter is optimized (denotes as L/R^*), the optimization moves to the next parameter, N_r, and the previous parameter L/R will be fixed at the optimized value L/R^* (Eqs. (8)–(10)). Each parameter is optimized by one or several iterations. During each iteration, parameter values (L/R and N_r) are selected with a specific step size within a specific range until the optimal parameter value (L/R^* and $N_r{}^*$) is found. Iteration with a smaller specific step size and a more specific range than the previous iteration can be conducted until the normalized absolute value (ε) is satisfied. Normalized absolute value is defined as the absolute difference between the target value ($\widetilde{R_r}$ and $\tilde{\vartheta}$) and optimized value (L/R^* and $N_r{}^*$). The absolute difference is then normalized by the target value ($\widetilde{R_r}$ and $\tilde{\vartheta}$). The parameter values are optimized when the normalized absolute values fall within the tolerance factor of 0.02 ($\varepsilon < 0.02$). Since finding the best optimization process is not in our focus, the specific step size is chosen after testing a set of arbitrary step sizes. The simulation consists of two stages, where the results for the initial optimization of the L/R ratio is shown in (Fig. 8(B) and (C)) followed by the optimization of N_r (Fig. 8(E) and (F)).

Firstly, we generated origami structures with different L/R values (Fig. 8(B) and (C)). The L/R values are selected from the range of [1, 1/2.25], with a step size of +0.25 being applied to the denominator. The R_r of each folded origami was measured using ImageJ, and we found that the targeted R_r (0.792) fell between L/R [1/1.50, 1/1.75] (Fig. 8(C)). Since the normalized absolute difference (ε) between the AGS simulated R_r (0.801) when 1/1.50 and targeted R_r (0.792) is smaller than 0.02, the stopping rule was satisfied (Fig. 8(C)). Thus, L/R 1/1.50 is identified as the optimal parameter value and was used in the following optimization process.

Optimization of R_r was followed by the optimization of the azimuthal angle (Fig. 8(E) and (F)). A similar procedure was conducted by generating different N_r selected from the range of [3,18] with a step size of +3. The targeted azimuthal angle ($\vartheta = 540°$) fell between N_r [9,12]. Neither $N_r = 9$ ($\vartheta = 482.028°$) nor $N_r = 12$ ($\vartheta = 665.075°$) satisfied the stopping rule (Fig. 8(F)). Hence, we conducted another round of optimization that is specific to the range of [9,12], with a step size of +1. The optimization process ended after the second iteration, since $N_r = 10$, the normalized absolute difference between the desired azimuthal angle ($\vartheta = 540°$) and the system-generated azimuthal angle (($\vartheta = 541.726°$) is smaller than 0.02 (satisfying the stopping rule).

Both optimal parameter values ($L/R = 1/1.50$, $N_r = 10$) are used to physically reconstruct the optimal spring origami structure. The system-optimized origami was physically folded and was physically tested in a cylindrical tube with a 1.9 cm inner radius r_1 and 4.3 cm outer radius r_2. The folded origami was able to twine with 1.925 cm inner radius r_1, 4.356 cm outer radius r_2 and 562.087° azimuthal angle in the preset cylindrical tube (Fig. 8(G) and (H)). The normalized absolute differences of the R_r and the azimuthal angle between physical folded prototype and the targeted origami requirements are both smaller than 4%. Further optimization studies can be conducted using other methods such as the Gradient Descent optimization method to provide a better optimization result [50].

To further capitalize on the advantages of our system design optimization, further studies can integrate an automated measurement system to fully automate the optimization process. Reinforcement machine learning can be incorporated with the automated optimization process to generate a fully automated spring origami designing pipeline that can generate spring origami with the desired requirements without any help provided by the user [51].

3.4. A time-saving origami design system

We illustrate our system as a time-saving procedure by comparing the total time needed to achieve the optimal design requirements shown in the previous section. We used the same optimization method, which provides similar attempts to achieve the same optimal design requirements with the same stopping rule. Similar L/R and N_r are chosen followed by the measurements of the R_r values and azimuthal angles (Fig. 8(C) and (F) labelled in blue). After the optimizations for both parameters, physical optimization showed identical optimal parameter values as system optimization (Fig. 8(C) and (F) labelled in blue), i.e. $L/R = 1/1.50$, $N_r = 10$. Since both optimization methods can provide identical optimal parameter values and our validation studies show that system folding can simulate similar physical folding behaviours in some perspectives (refer back to the validation section), we suggest replacing physical optimization with system optimization to resolve the time-consuming issue during design optimization. The major time-consuming steps are generating the 2D origami structure and the physical folding of the origami structure. A study was conducted to show the average time spent generating and folding the origami in both validation methods (Table 2). The time spent for both methods are shown below.

Table 2
Average time spent for different steps and optimization methods.

	Physical	Auto-generating system
Generating step	6 min/origami	<30 s/origami
Folding step	10 s/crease	<30 s/batch
Total optimization time (exclude measurements)	649.5 min	15 min

Table 2 shows the average time spent, taken from three independent measurements. Results show that physical optimization is 43.3 times longer than Auto-generating System optimization. In addition, system optimization provides a convenient approach that enables the user to generate multiple origami structures and fold multiple origami structures simultaneously (Supplementary S1 and Supplementary S2). In contrast, physical optimization can only generate and fold one origami structure at any one point. In addition, manual-folding of origami usually requires pre-creasing with the folding occurring sequentially, whereas the simulation does all folds simultaneously. Currently, the measurement step of both optimization processes is done manually. Further studies can develop an automated measurement step to fully automate the system optimization process (generating origami, folding origami and taking measurements).

3.5. Feasibility of the system and the choice of platform

Asymmetrical origami spring structures can also withstand a load exerted on them in a certain direction (Fig. 9). As shown in Fig. 10, there are similar behaviours for symmetrical origami spring structures in both the computational output and the physical folding. Observations suggest the AGS simulated-symmetrical origami structure to be flat foldable, agreeing with the physical prototype (Fig. 10(A)). In contrast, the physical asymmetrical origami structure is observed to give rise to a structure that is not flat foldable and is capable of directionally withstanding force (Fig. 10(B)). However, due to its algorithm, AGS is not able to detect the collisions and hence will fully compress the origami structure with collision (Figs. 10(B) and 11(A)–(D)). On the other hand, fully compressed origami with collision is not achievable in physical folding. This illustrates that some AGS simulated output may not be feasible in physical folding due to the inability of AGS to detect the collision during folding.

The symmetrical origami structure is observed to be flat foldable. In contrast, the asymmetrical origami structure is observed to give rise to a structure that is not flat-foldable and is capable of directionally withstanding force (Fig. 9). However, due to the algorithm in AGS, the simulated origami structure demonstrates collision within the folded structure to achieve flat-foldability (Fig. 10(B)), which is not achievable in physical folding. This limitation can be mitigated by using different origami folding platforms to avoid collisions. As shown in

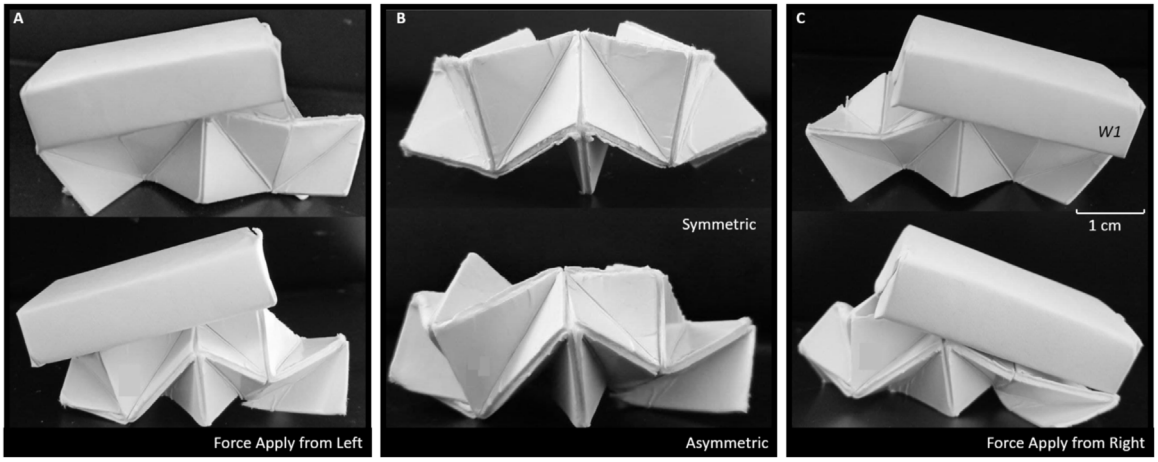

Fig. 9. Comparison of symmetrical and asymmetrical origami spring structures when withstanding a load of 70 g (denote as W1) exerted on it from a certain direction (either left side or right side). Asymmetric origami spring structures can only withstand the load from one direction and collapse in the other, and the direction which can withstand the force depends on the L/R ratio. (A) The force is applied from the left, and both the symmetric and asymmetric origami spring structures can withstand the load. (B) Unloaded state of the symmetric and asymmetric structure. (C) The force is applied from the right, the symmetric structure can withstand the force, but the asymmetric structure collapses.

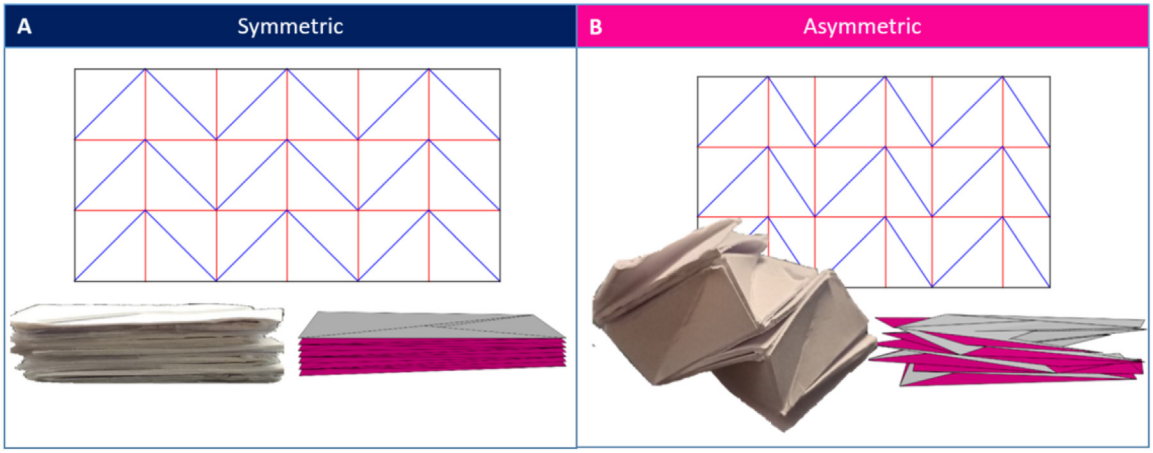

Fig. 10. Comparison of manual and physical folding with computational folding for both symmetric and asymmetric structures. 2D origami spring structure (upper), physical folding (bottom left) and computational folding (bottom right) are displayed. (A) Symmetric origami spring structure: The simulation of symmetric spring origami agrees with the physical prototype because both types of folding can fully compress. (B) Asymmetric origami spring structure: The simulation of asymmetric origami is different from the asymmetric physical prototype because physical folding cannot fully compress, while computational folding can fully compress with collision. However, fully compressing with collision is not feasible in reality. This illustrates that some AGS simulated output may not be feasible in physical folding.

(Fig. 11(C)–(E)), when using an identical structure (identical output file with different file types), collisions are observed in AGS (Fig. 11(C)–(D)). However, collisions are avoided in Freeform Origami folding (Fig. 11E).

The output of the proposed system is an output file that consists of single/multiple origami structures. Different computational folding platforms (such as AGS and Freeform Origami) have different limitations. Thus, interchanging between different platforms not only provides a feasible way to overcome limitations, it can also be suited to different research purposes. Interchanging between different platforms can be done by converting the output file type (SVG file type) to the input file type required by the desired platform.

3.6. Strain analysis of spring origami structure

AGS allowed the comparison between symmetric and asymmetric spring origami structures in terms of origami strain. Furthermore, AGS can display different strain visualization by changing the maximum number of strains of the simulator. For instance, the simulator translates the strain into a RGB colour on the spectrum of blue (no strain) to red (maximum strain) and applies it to the 3D model for visualization [5]. This can be seen in Fig. 12(A). Strain visualization was generated by specified folding parameters. These parameters included maximum strain level, fold stiffness, axial stiffness, face stiffness, facet crease stiffness and damping ratio.

Strain analysis of twisting (Section 3(A)) and closing angle (Section 3(B)) origami structures were conducted under the same fold percentage (60%) with the maximum strains of 2.3% and 5%, respectively. The results are shown in Fig. 12(B) and (C), respectively. The results reveal symmetrical and asymmetrical strain distribution on symmetrical and asymmetrical origami structures, respectively (Fig. 12).

Results for twisting origami structure shows (Fig. 12(B)):

- When $H/L < H/R$ i.e., $\frac{H/R}{H/L} = L/R > 1$, the magnitude of strain level is proportional to $\frac{H/R}{H/L}$.
- When $H/L > H/R$, i.e., $\frac{H/R}{H/L} = L/R < 1$, the magnitude of strain level is proportional to $\frac{H/L}{H/R}$.
- When $H/L = H/R$, i.e., $\frac{H/R}{H/L} = L/R = 1$, the magnitude of strain level is the least.

From Fig. 12(C), we can observe the strain level of the closing angle shows:

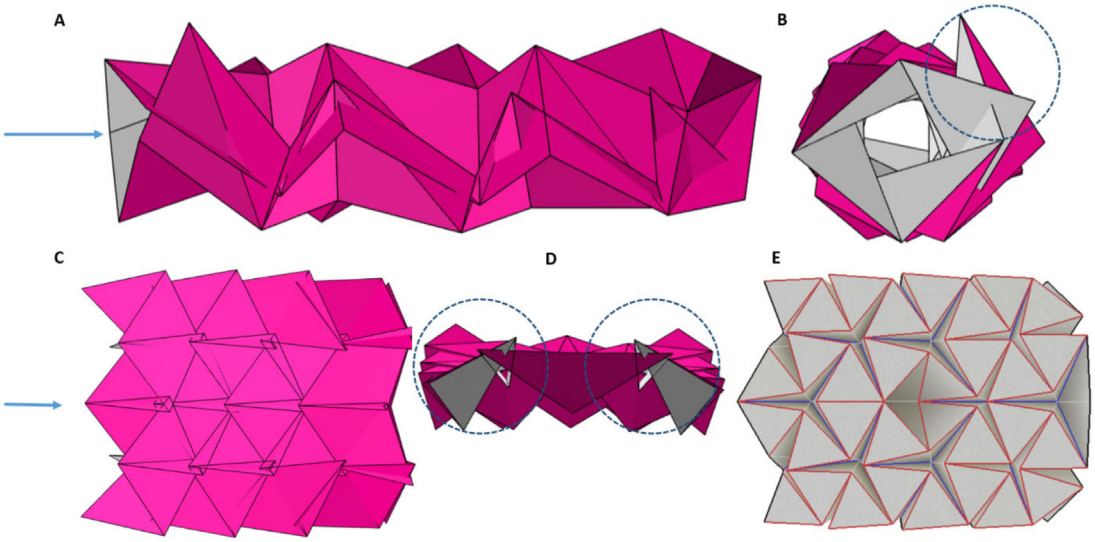

Fig. 11. Collision in AGS (A–D) and collision-free in Freeform Origami (E). (A)(C) Side view of the collision folding structure shows that the creases overlay, which is not feasible in reality. (B)(D) Front view of the collision folding structure (viewer's view from the direction of the arrow) shows that the creases intersect, which is not feasible in physical folding. (E) Identical origami structure used in (C) folding in Freeform Origami platform with collision avoiding function.

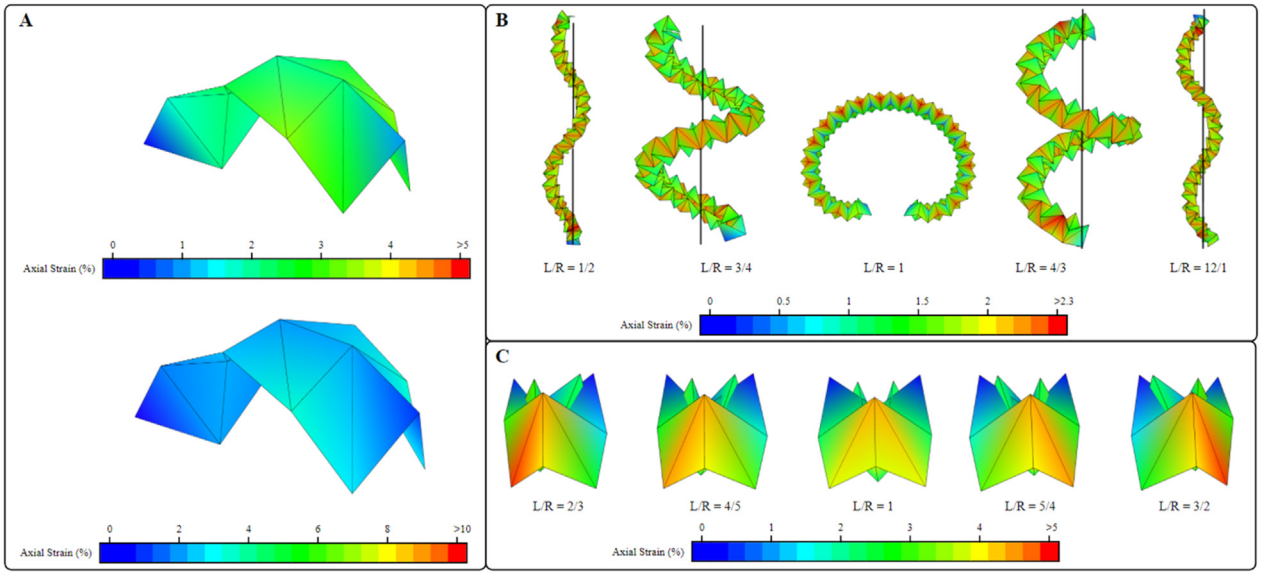

Fig. 12. Strain analysis result of twisting and closing angle of spring origami structures. Strain levels and strain distribution profiles vary directly as the L/R values of the spring origami structures. (A) Strain visualization of the side view of the spring origami structure with different maximum strain percentages. Maximum strain specified at 5% and 10%, respectively. (B) Shows the strain visualization of the twisting spring origami structures with 2.3% maximum strain, L/R values closer to one, the lower the strain levels are. (C) Shows the strain visualization of the closing angle spring origami structures with 5% maximum strain. Closer the L/R values to one, lower the strain levels and more centred strain distribution profiles.

- When $H/L < H/R$, i.e., $\frac{H/R}{H/L} = L/R > 1$, the strain distribution is shifted to the right. The strain level and the magnitude of the shift are proportional to $\frac{H/R}{H/L}$.
- When $H/L > H/R$, i.e., $\frac{H/R}{H/L} = L/R < 1$, the strain distribution is shifted to the left. The strain level and the magnitude of the shift are proportional to $\frac{H/L}{H/R}$.
- When $H/L = H/R$, i.e., $\frac{H/R}{H/L} = L/R = 1$, the strain distribution profile is symmetrical and centred. The strain level is the least.

Quantitative analysis of the strain level can be achieved by AGS website debugging to retrieve the specific numerical strain levels of the interested regions, which the AGS uses to generate the strain visualization.

4. Discussion and conclusion

Origami robotics can integrate computational design practices to speed up the research and development process using parametric dependencies [52,53]. Computational design can discover different origami patterns and useful folding mechanisms quickly. This study developed an auto-generating system that can be used with available computational software for a more efficient design process. The system's performance is demonstrated by taking a spring origami crease pattern as an example, and varying different parameters of the pattern, such as ratio, rows, repeats and opacity, and comparing the outputs. Different parameters and values can be introduced to generate different crease patterns. The output crease patterns can then serve as inputs for currently available computational origami software (AGS in this case)

to filter and extract the useful folding patterns, which can be used in real-life applications. After filtering the possibilities to obtain the few that seem to have promising results, the researcher can conduct physical folding experiments on the useful folding patterns to ensure the feasibility of the patterns in real-life applications.

According to the experiments conducted in this study, a small difference in parameters (especially the horizontal length ratios) will lead to very different spring origami folding mechanisms. Asymmetricity causes asymmetric features such as twisting, closing angle and directional load-bearing in spring origami units. In particular, similar origami features (e.g., twisting and closing angle) can be observed in both computational origami and physical folding. Additionally, when compared against physical folding, computational folding produced similar trends (e.g., magnitude of twisting and closing angle). However, as AGS cannot prevent collisions, it will lead to unfeasible simulated folding of spring origami structures which cannot be performed in real life. This problem often occurs in folding experiments for asymmetric structures [5], as the AGS can only fold uniform crease patterns. Localized and varying degrees of folding of different areas cannot be performed using the AGS. As such, the consequent features (e.g., directional load-bearing) will not be observable computationally. However, the online computational folding platform can serve as a reference to the overall folding profile. Comparison between physical folding and AGS can ensure that the prediction of the folding mechanism is correct.

Through the simultaneous input and output of the system, this system can be a part of push-button design automation, enabling close human–robot interaction robots to adapt and respond to environmental changes [54] simultaneously. The system potentially automates the entire process from design to manufacturing and collapses the two into one [54,55]. This design can be implemented in the tessellated crease pattern synthesis used in origami-guided and shelled soft robots. Using specific parameters, desired kinematics (such as twisting direction and closing angle) can be specified for origami-guided and shelled soft robots [55–57]. Future studies can utilize the inverse kinematics method to decompose desired Cartesian coordinates and transform them into specific origami guided/shell robotic joint angles to achieve desired folding patterns via our system [4,58,59] and multiple sensing modalities [60].

Abbreviation

Eos	E-origami system
AGS	Amanda Ghassei Simulator
SVG	Scalable Vector Graphics
H	Height of the spring origami unit
L	Horizontal base length for left part of spring origami unit
R	Horizontal base length for right part of spring origami unit
M_r	Number of rows in the spring origami structure
N_r	Number of repeats in the spring origami structure
Y_{OA}	The difference of y-axis coordinate between the origin and the right hinge of the single origami base
Y_{OB}	The difference of y-axis coordinate between the origin and the left hinge of the single origami base
R_r	The normalized radius of the spiralling origami
r_1	Inner radius of the cylindrical tube
r_2	Outer radius of the cylindrical tube
ϑ	Azimuthal angle
ϵ	Normalized absolute value

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank Xin Yi Than for the help with the origami folding and data collection; Ryan Lee Ting Zhern for his advice and help in the optimization experiments.

This work was supported by the Chinese University of Hong Kong (CUHK) Direct Grant (4055139) for a research project on Multiphysics Study of Magnetically Deployable Robotic Collapsible Structures.

Appendix A. Supplementary data

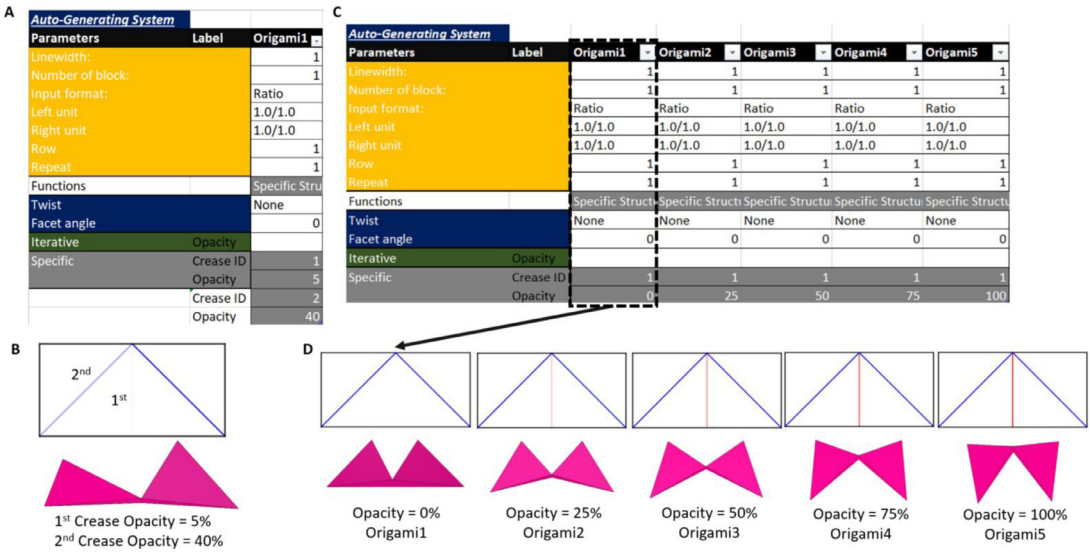

Fig. S1. Illustration of Specific Function. (A) User input for generating a modified origami structure. The vertical mountain crease (crease ID = 1, labelled as "1st" crease in subfigure B) was modified with 5% crease opacity and the diagonal valley crease (crease ID = 2, labelled as "2nd" crease in subfigure B) was modified with 40% crease opacity. (B) The corresponding 2D origami structure and the 3D AGS folding in 60 fold percentage. (C) User generates five origami structures with Specific Function. The vertical mountain crease (crease ID = 1) in each origami structure contains different crease opacities (0%, 25%, 50%, 75% and 100%). (D) Each column (dotted square in (C)) is respected to one origami structure in (D). The system generates five origami structures with respective crease opacity and the corresponding structures were folded in AGS with a 60 fold percentage. As mentioned, different crease opacity can have a different folding magnitude in the AGS. The "seat number" is called "crease ID" in the user interface. The details of the "seat number" assignment are mentioned in the Excel template.

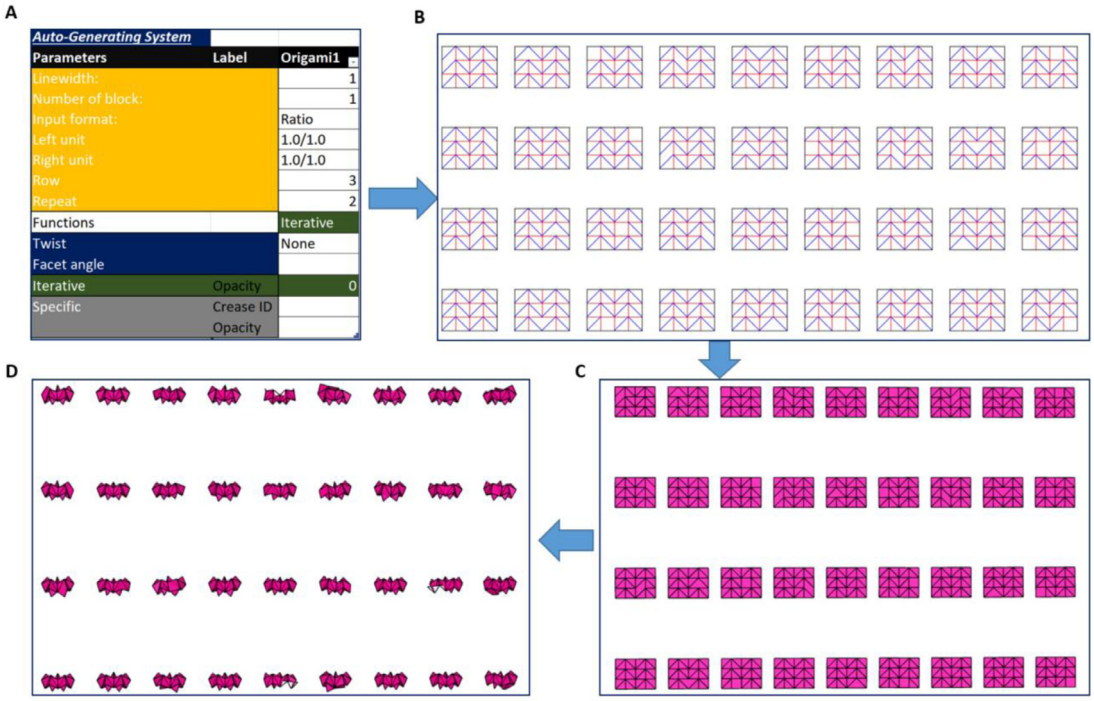

Fig. S2. Illustration of Iteration Function. (A) User chooses 0% crease opacity to generate an origami structure consisting of three rows and two repeat units (36 creases) using the iterating function. (B) The system generates 36 derivative structures in a single output file. Each structure has one crease with 0% crease opacity, with none of them having the same transparent crease. (C) The unfolded derivative structures were subjected to AGS. (D) All the derivative structures were folded simultaneously in AGS with a 60 fold percentage. This function enables the mass comparison of the actuated derivative origami structure with the different modified creases.

References

[1] R.J. Lang, A computational algorithm for origami design, in: Proc. 12th Annual ACM Symp. on Computational Geometry, 1996, pp. 98–105, http://dx.doi.org/10.1145/237218.237249.

[2] R.J. Lang, Origami 4, A K Peters/CRC Press, New York, 2009.

[3] L. Hardesty, New Algorithm Generates Practical Paper-Folding Patterns to Produce Any 3-D Structure, MIT News Office, 2017.

[4] D. Rus, T.M. Tolley, Design, fabrication and control of origami robots, Nat. Rev. Mater 3 (6) (2018) 101–112, http://dx.doi.org/10.1038/s41578-018-0009-8.

[5] A. Ghassaei, Fast, interactive origami simulation using GPU computation, 2018.

[6] A. Kasem, T. Ida, H. Takahashi, M. Marin, F. Ghourabi, Eorigami system Eos, in: Proceedings of the Annual Symposium of Japan Society for Software Science and Technology (JSSST, Tokyo, Japan), 2006, http://dx.doi.org/10.1201/b10653-29.

[7] T. Ida, H. Takahashi, M. Marin, F. Ghourabi, Modeling origami for computational construction and beyond, in: O. Gervasi, M.L. Gavrilova (Eds.), Computational Science and its Applications – ICCSA 2007. ICCSA 2007, in: Lecture Notes in Computer Science, vol. 4706, 2007, http://dx.doi.org/10.1007/978-3-540-74477-1_60.

[8] R.J. Lang, Treemaker 4.0: A program for origami design, 1998, https://langorigami.com/article/treemaker/ (accessed 20 May 2021).

[9] R.J. Lang, Origami Design Secrets, second ed., 2011.

[10] K. Kuribayashi, T. Kiochi, Z. You, T. Dacian, U. Minoru, I. Takahiro, S. Masahiro, Self-deployable origami stent grafts as a biomedical application of Ni-rich TiNi shape memory alloy foil, Mater. Sci. Eng. 419 (1–2) (2006) 131–137, http://dx.doi.org/10.1016/j.msea.2005.12.016.

[11] T. Tachi, Freeform origami, 2020, http://www.tsg.ne.jp/TT/software/ (Accessed Dec 10, 2020) June 16.

[12] T. Tachi, Generalization of rigid-foldable QuadrilateralMesh origami, J. Int, Assoc. Shell Spat. Struct. (IASS) 50 (3) (2009) 173–179, ISBN: 978-84-8363-461-5.

[13] T. Tachi, Freeform Variations of Origami, in: Proceedings of the 14th International Conference on Geometry and Graphics (ICGG 2010) 2010, pp. 273-274.

[14] T. Tachi, Freeform rigid-foldable structure using bidirectionally flat-foldable planar quadrilateral mesh, Adv. Archit. Geom. (2010) 87–102, http://dx.doi.org/10.1007/978-3-7091-0309-8_6.

[15] T. Tachi, Origamizer, 2020, http://www.tsg.ne.jp/TT/software/ (Accessed Dec 10, 2020) June 16.

[16] T. Tachi, Origamizing polyhedral surfaces, IEEE Trans. Vis. Comput. Graphics 16 (2) (2010) 298–311, http://dx.doi.org/10.1109/TVCG.2009.67.

[17] T. Tachi, 3D Origami Design Based on Tucking Molecule, Vol. 4, Origami, 2009, pp. 259–272.

[18] E. Hawkers, B. An, N.M. Benbernou, H. Tanaka, S. Kim, E.D. Demaine, D. Rus, R.J. Wood, Programmable matter by folding, Proc. Natl. Acad. Sci. 107 (28) (2010) 12441–12445, http://dx.doi.org/10.1073/pnas.0914069107.

[19] Y.L. Cheng, J.H. Scott, Computational design of tissue engineering scaffolds, Comput. Methods Appl. Mech. Engrg. 196 (31–32) (2007) 2991–2998.

[20] Caio C. Lucarelli, Joyce, Parametric modeling simulation for an origami shaped canopy, Front. Archit. Res. (2019) http://dx.doi.org/10.1016/j.cma.2006.09.023.

[21] C.J. Cai, et al., Diversified and untethered motion generation via crease patterning from magnetically actuated caterpillar-inspired origami robot, IEEE/ASME Trans. Mechatronics (2020) http://dx.doi.org/10.1109/TMECH.2020.3028746.

[22] N. Wonoto, B. Daniel, G. Russell, S. Matthew, Parametric design and structural analysis of deployable origami tessellation, Comput.-Aided Des. Appl. 10 (6) (2013) 939–951, http://dx.doi.org/10.3722/cadaps.2013.939-951.

[23] R.K. Katzschmann, A.D. Marchese, D. Rus, Autonomous object manipulation using a soft planar, Soft Robot. 2 (2015) 4, http://dx.doi.org/10.1089/soro.2015.0013.

[24] A.D. Marchese, R.K. Katzschmann, D. Rus, A recipe for soft fluidic elastomer robots, Soft Robot. 2 (2015) 1, http://dx.doi.org/10.1089/soro.2014.0022.

[25] C.D. Onal, R.J. Wood, D. Rus, An origami-inspired approach to worm robots, IEEE/ASME Trans. Mechatronics 18 (2013) 2, http://dx.doi.org/10.1109/TMECH.2012.2210239.

[26] S. Ornes, Inner workings: Medical microrobots have potential in surgery, therapy, imaging, and diagnostics, Proc. Natl. Acad. Sci. USA 114 (47) (2017) 12356–12358, http://dx.doi.org/10.1073/pnas.1716034114.

[27] B. Zhang, et al., Worm-like soft robot for complicated tubular environments, Soft Robot. 3 (2019) 6, http://dx.doi.org/10.1089/soro.2018.0088.

[28] B. Chen, et al., Soft origami gripper with variable effective length, Adv. Intell. Syst. (2021) http://dx.doi.org/10.1002/aisy.202000251.

[29] A. Kanada, et al., Reachability improvement of a climbing robot based on large deformations induced by tri-tube soft actuators, Soft Robot. 6 (2019) 4, http://dx.doi.org/10.1089/soro.2018.0115.

[30] S. Chandra, A. Körner, A. Koronaki, R. Spiteri, R. Amin, S. Kowli, M. Weinstock, Computing curved-folded tessellations through straight folding approximation, in: Proc. Symposium on Simulation for Architecture & Urban Design, 2015, http://dl.acm.org/citation.cfm?id=2873042.

[31] M. Achim, A. Sean, Computational Design Thinking, AD Reader, ISBN: 978-0-470-66570-1, 2011.

[32] C. Bohan, Soft origami gripper with variable effective length, Adv. Intell. Syst. (2021) 3, http://dx.doi.org/10.1002/aisy.202000251.

[33] C. Ai, Current development on origami/kirigami-inspired structure of creased patterns toward robotics, Adv. Energy Mater. 23 (10) (2021) http://dx.doi.org/10.1002/adem.202100473.

[34] H. Fang, Y. Zhang, K.-W. Wang, Origami-Based Earthworm-Like Locomotion Robots, IOPscience, 2017, http://dx.doi.org/10.1088/1748-3190/aa8448.
[35] M. Sivaperuman Kalairaj, C.J. Cai, S. Pavitra, H. Ren, Untethered origami worm robot with diverse multi-leg attachments and responsive motions under magnetic actuation, Robotics 2021 10 (118) (2021) http://dx.doi.org/10.3390/robotics10040118.
[36] J.-E. Suh, T.-H. Kim, J.-H. Han, New approach to folding a thin-walled yoshimura patterned cylinder, J. Spacecr. Rockets 58 (2021) 2, http://dx.doi.org/10.2514/1.A34784.
[37] J.S. Koh, K.J. Cho, Omegabot: Biomimetic inchworm robot using SMA coil actuator and smart composite microstructures (SCM), in: IEEE Int. Conf. Robot. Biomimetics, ROBIO 2009, 2009, pp. 1154–1159, http://dx.doi.org/10.1109/ROBIO.2009.5420752.
[38] H. Fang, Y. Zhang, K.W. Wang, Origami-based earthworm-like locomotion robots, Bioinspir. Biomim. 12 (6) (2017) 65003, http://dx.doi.org/10.1088/1748-3190/aa8448.
[39] H. Fang, Y. Zhang, K.W. Wang, An earthworm like robot using origami-ball structures, in: Active and Passive Smart Structures and Integrated Systems 2017, Vol. 10164, 2017, p. 1016414, http://dx.doi.org/10.1117/12.2258703.
[40] C.D. Onal, R.J. Wood, D. Rus, An origami inspired approach to worm robots, IEEE/ASME Trans. Mechatronics 18 (2) (2013) 430–438, http://dx.doi.org/10.1109/TMECH.2012.2210239.
[41] Q. Zhang, H. Fang, J. Xu, Yoshimura-origami based earthworm-like robot with 3-dimensional locomotion capability, Front. Robot. AI 8 (2021) 738214, http://dx.doi.org/10.3389/frobt.2021.738214.
[42] M. Luo, R. Yan, Z. Wan, Y. Qin, J. Santoso, E.H. Skorina, C.D. Onal, Orisnake: Design, fabrication and experimental analysis of a 3-D origami snake robot, IEEE Robot. Autom. Lett. (1993) http://dx.doi.org/10.1109/LRA.2018.2800112.
[43] A. Pagano, B. Leung, B. Chien, T. Yan, A. Wissa, S. Tawfick, Multi-Stable Origami Structure for Crawling Locomotion, American Society Of Mechanical Engineers, Stowe, Vermont, USA, 2016, V002T06A005, http://dx.doi.org/10.1115/SMASIS2016-9071.
[44] G. Curletto, Rigid foldable origami structures: parametric modelling with grasshopper, in: Geometric and Structural Issues, Lisbon, Portugal, s.n, 2016.
[45] B. Douglas, J.C. Anne, Innovative uses of video analysis, Phys. Teach. 47 (2009) 145–150, http://dx.doi.org/10.1119/1.3081296.
[46] C.A. Schneider, W.S. Rasband, K.W. Eliceiri, NIH Image to imagej: 25 years of image analysis, Nature Methods 9 (7) (2012) 671–675, http://dx.doi.org/10.1038/nmeth.2089.
[47] M. Yang, et al., Twining plant inspired pneumatic soft robotic spiral gripper with a fiber optic twisting sensor, Opt. Express 28 (2020) 35158–35167, http://dx.doi.org/10.1364/OE.408910.
[48] S.J. Wright, Coordinate descent algorithms, Math. Program. 151 (1) (2015) 3–34, http://dx.doi.org/10.1007/s10107-015-0892-3.
[49] W. Kahan, Gauss–Seidel Methods of Solving Large Systems of Linear Equations (Ph.D. thesis), University of Toronto, Toronto, Canada, 1958.
[50] C. Lemaréchal, Cauchy and the Gradient Method, Doc Math Extra, 2012, pp. 251–254.
[51] P. Rajak, B. Wang, Nomura. Ki, et al., Autonomous reinforcement learning agent for stretchable kirigami design of 2D materials, Npj Comput. Mater. 7 (2021) 102, http://dx.doi.org/10.1038/s41524-021-00572-y.
[52] S. Carta, Machine learning and computational design, 2020, http://dx.doi.org/10.1145/3401842.
[53] M. Johnson, C. Yue, H. Sierra, X. Sheng, W. Bradford, R. Hongliang, T. Junichi, T.H.T. Zion, Fabricating biomedical origami: a state-of-the-art review, Int. J. Comput. Assist. Radiol. Surg. 12 (2017) 2023–2032, http://dx.doi.org/10.1007/s11548-017-1545-1.
[54] J. Paik, Soft robot design methodology for 'push-button' manufacturing, Nat. Rev. Mater. 3 (2018) 81–83, http://dx.doi.org/10.1038/s41578-018-0014-y.
[55] L. Paez, A. Gunjan, P. Jamie, Design and analysis of a soft pneumatic actuator with origami shell reinforcement, Soft Robot. 3 (3) (2016) 109–119, http://dx.doi.org/10.1089/soro.2016.0023.
[56] W. Kim, B. Junghwan, K. Jae-Kyeong, C. Woo-Young, J. Kirsten, J. Joachim, L. Dae-Young, C. Kyu-Jin, Bioinspired dual-morphing stretchable origami, Science Robotics 4 (36) (2021) http://dx.doi.org/10.1126/scirobotics.aay3493.
[57] L. Wang, S. Wei-Li, F. Daining, Twistable origami and kirigami: from structure-guided smartness to mechanical energy storage, ACS Appl. Mater. Interfaces 11 (3) (2019) 3450–3458, http://dx.doi.org/10.1021/acsami.8b17776.
[58] T. Tachi, Geometric considerations for the design of rigid origami structures, in: International Association for Shell and Spatial Structures (IASS) Symposium, Shanghai, 2010.
[59] E.R. Leal, S.D. Jian, From origami to a new class of centralized 3-DOF parallel, in: ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 8, 2007, pp. 1183–1193, http://dx.doi.org/10.1115/DETC2(Curletto,2016)007-35516.
[60] H. Ren, P. Kazanzides, Investigation of attitude tracking using an integrated inertial and magnetic navigation system for hand-held surgical instruments, IEEE/ASME Trans. Mechatron. 17 (2) (2012) 210–217, http://dx.doi.org/10.1109/TMECH.2010.2095504.