EPiC Series in Computing
Volume 39, 2016, Pages 68–80
SCSS 2016. 7th International Symposium on Symbolic Computation in Software Science

A Mathematica module for conformal geometric algebra and origami folding

Mitsuhiro Kondo[1]*, Takuya Matsuo[1], Yoshihiro Mizoguchi[2], and Hiroyuki Ochiai[2]

[1] Graduate School of Mathematics, Kyushu University, Japan
[2] Institute of Mathematics for Industry, Kyushu University, Japan

Abstract

We implemented a *Mathematica* module for conformal geometric algebra (CGA) which includes functions to represent CGA elements and compute operations on the elements. In particular, we can draw a figure in 3D space corresponding to a CGA element. The proposed drawing function uses a Gröbner basis for simplifying the corresponding equations. This function can visualize any CGA element. One motivation of the present study is to realize a 3D origami system using our own CGA library. This 3D system is based on the 2D computational origami system E-Origami-System developed by Ida et al. and simple fold operations were formulated in 3D using CGA points and motions. We then prove geometric theorems concerning 3D origami properties using the proposed module.

1 Introduction

Complex numbers can be used to describe a point on a plane. The addition of complex numbers can be considered to be a translation of a point on a plane, while their multiplication can be considered to be a magnification, reduction, and rotation transformation of a point on a plane. Quaternions are a number system which is an extension of the complex numbers introduced by W.R. Hamilton in 1843. We can use the quaternion numbers to describe a point in a three-dimensional (3D) space. Calculation of a quaternion involves a rotation of a point in 3D space. Since it is easy to manipulate a computation using its algebraic properties, quaternions are used in applied mathematics, e.g., in 3D computer graphics and computer vision [10, 7, 11]. Conformal geometric algebra (CGA) is an extension of algebraic number systems which can involve space points, rotations, magnifications, reductions, and translations of higher-dimensional spaces [1, 12, 9].

In the present paper, we investigate CGA to consider figures in $\mathbb{R}^3$ and their transformations. We herein consider a CGA as a 32-dimension linear space. Transformations of figures in $\mathbb{R}^3$ can be described using a 4×4 affine transformation matrix, and a point in a figure can be described using a vector in $\mathbb{R}^3$. Our motivation for using CGA is to describe both points and transformations in a single algebraic framework in order to manipulate the computation and to

*kondo.mitsuhiro@kyudai.jp

easily verify its properties. One of our research goals is to find a useful algorithm for visualizing the basic geometric entities used in computer graphics through the concepts in CGA.

We implemented a *Mathematica* module which includes functions to represent CGA elements and to compute operations in CGA, such as the geometric product, the inner product, and the outer product. Furthermore, we can draw the figure in $\mathbb{R}^3$ which corresponds to a CGA element. Our drawing function uses a Gröbner basis for simplifying equations corresponding to figures. Moreover, our drawing function can also be used for multi-dimensional figures. We also implemented a function to check the equality of two figures which are represented by different CGA elements [8].

Some origami are made in 2D, as shown in Figure 1. Some of these origami, such as the crane origami, can be extended to 3D after all folding operations, but these are closed in 2D during folding. Since all fold operations are π folding, 2D origami calculation is closed in 2D. However, there are some origami which cannot be obtained by π folding alone. For example, in order to construct a piano, as shown in Figure 2, we need to use $\frac{\pi}{2}$ folding. We then consider an application of CGA to the 3D origami formalization and investigate its properties.

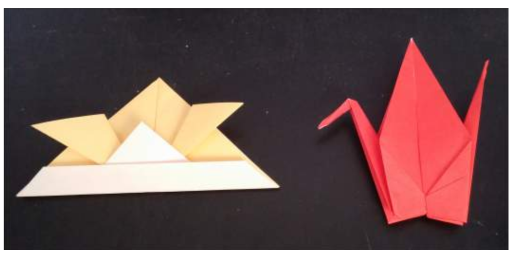

Figure 1: 2D Origami

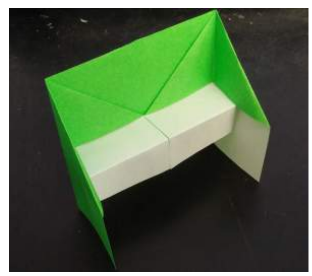

Figure 2: 3D Origami

Ida et al. formalized 2D origami development using the system Eos and verified a number of origami properties [3, 6, 5, 2]. In 2014, Ida introduced the concept of extending Eos to 3D origami [4]. Our goal is to realize a 3D origami system using our own CGA library. Based on the Eos 2D computational origami system, simple fold operations in 3D were formulated using CGA points and motions. We then proved a simple geometric theorem for 3D origami properties by calculating CGA equation formulas.

2 Conformal geometric algebra

Let $\mathcal{A}$ be the finite set $\{0, 1, 2, 3, \infty\}$. We define sets of variables $W = \{w_S \mid S \subset \mathcal{A}\}$ and $E = \{e_S \mid S \subset \mathcal{A}\}$. A geometric algebra G has six operators, Product ($* : G \times G \to G$), Sum ($+ : G \times G \to G$), Minus ($- : G \to G$), Outer product ($\wedge : G \times G \to G$), Inner product ($\cdot : G \times G \to G$), and Scalar product ($\cdot : \mathbb{R} \times G \to G$). In order to describe an element of the geometric algebra G as a polynomial in $\mathbb{R}[E]$ or $\mathbb{R}[W]$, we define some computing functions on $\mathbb{R}[E]$ and $\mathbb{R}[W]$. We assume that $w_{\{a\}} = e_{\{a\}}$ $(a \in \mathcal{A})$, and

$$e_{\{i\}} \cdot e_{\{j\}} = \begin{cases} 1 & (i = j \wedge (1 \leq i, j \leq 3)) \\ 0 & (i \neq j \wedge (1 \leq i, j \leq 3)) \vee (i = j = 0) \vee (i = j = \infty) \\ -1 & ((i = 0 \wedge j = \infty) \vee (i = \infty \wedge j = 0)). \end{cases}$$

In order to compute a product of two polynomials in $\mathbb{R}[E]$, we define

$$\begin{aligned}
e_S * e_\phi &= e_\phi * e_S = e_S, \\
e_{\{0\}} * e_S &= \begin{cases} e_{\{0\}\cup S} & (0 \notin S) \\ 0 & (0 \in S), \end{cases} \\
e_{\{a\}} * e_S &= \begin{cases} (-1)^{|\{s\in S|s<a\}|}\, e_{\{a\}\cup S} & (a \in \{1,2,3\} \wedge a \notin S) \\ (-1)^{|\{s\in S|s<a\}|}\, e_{S-\{a\}} & (a \in \{1,2,3\} \wedge a \in S), \end{cases} \\
e_{\{\infty\}} * e_S &= \begin{cases} (-1)^{|S|}\, e_{\{\infty\}\cup S} & (0 \notin S \wedge \infty \notin S) \\ 0 & (0 \notin S \wedge \infty \in S) \\ -e_{\{0\}} * (e_{\{\infty\}} * e_{S-\{0\}}) - 2e_{S-\{0\}} & (0 \in S), \end{cases} \\
e_{T\cup\{a\}} * e_S &= e_T * (e_{\{a\}} * e_S)\ (\forall t \in T, t < a).
\end{aligned}$$

For computing a product of two polynomials in $\mathbb{R}[W]$, we use the following transformation and computations in $\mathbb{R}[E]$.

$$w_S = \begin{cases} e_S & (0 \notin S \vee \infty \notin S) \\ e_S + (-1)^{|S|} e_{S-\{0,\infty\}} & (0 \in S \wedge \infty \in S). \end{cases}$$

We define the function $grade : \mathbb{R}[W] \to \mathbb{N}$ by $grade(aw_S) = |S|$ $(a \in \mathbb{R})$ and $grade(f+g) = max(grade(f), grade(g))$ $(f, g \in \mathbb{R}[W])$. For $k \in \mathbb{N}$, the function $pickup_k : \mathbb{R}[W] \to \mathbb{R}[W]$ identifies terms which have $grade(f) = k$ $(f \in \mathbb{R}[W])$. That is, $pickup_2(w_0+w_{01}+w_{012}+w_{12}) = w_{01} + w_{12}$. Note that $w_\phi, w_{\{0\}}$, $w_{\{0,1\}}$, and $w_{\{0,1,2\}}$ are denoted by $1, w_0$, w_{01}, and w_{012}, respectively. In order to compute an outer product($\wedge$) and an inner product($\cdot$) in $\mathbb{R}[W]$, we define

$$\begin{aligned}
w_S \wedge w_T &= pickup_{|grade(w_S)+grade(w_T)|}(w_S * w_T), \text{and} \\
w_S \cdot w_T &= pickup_{|grade(w_S)-grade(w_T)|}(w_S * w_T).
\end{aligned}$$

Note that $x * y = x \cdot y + x \wedge y$ for any elements $x, y \in \mathbb{R}[W]$. In Tables 1 and 2, we list the operation tables of the product and the outer product, respectively.

Table 1: Product operation table

$*$	e_0	e_1	e_2	e_3	e_∞
e_0	0	e_{01}	e_{02}	e_{03}	$e_{0\infty}$
e_1	$-e_{01}$	1	e_{12}	e_{13}	$e_{1\infty}$
e_2	$-e_{02}$	$-e_{12}$	1	e_{23}	$e_{2\infty}$
e_3	$-e_{03}$	$-e_{13}$	$-e_{23}$	1	$e_{3\infty}$
e_∞	$-2-e_{0\infty}$	$-e_{1\infty}$	$-e_{2\infty}$	$-e_{3\infty}$	0

$*$	w_0	w_1	w_2	w_3	w_∞
w_0	0	w_{01}	w_{02}	w_{03}	$-1+w_{0\infty}$
w_1	$-w_{01}$	1	w_{12}	w_{13}	$w_{1\infty}$
w_2	$-w_{02}$	$-w_{12}$	1	w_{23}	$w_{2\infty}$
w_3	$-w_{03}$	$-w_{13}$	$-w_{23}$	1	$w_{3\infty}$
w_∞	$-1-w_{0\infty}$	$-w_{1\infty}$	$-w_{2\infty}$	$-w_{3\infty}$	0

A function exp on a CGA is defined by a formal power series $\exp(x) = \sum_{k=0}^{\infty} \frac{1}{k!} x^k$ where we abbreviate a product operator symbol $*$. We note that $\exp(x) \in \mathbb{R}[W]$ for any $x \in \mathbb{R}[W]$.

There are known relationships between CGA elements and figures in $\mathbb{R}^3$. A *point* $(x, y, z) \in \mathbb{R}^3$ is represented by a CGA element $P_{(x,y,z)} = w_0 + xw_1 + yw_2 + zw_3 + \frac{1}{2}\left(x^2 + y^2 + z^2\right) w_\infty$.

Table 2: OuterProduct operation table

$\wedge$	e_0	e_1	e_2	e_3	e_∞
e_0	0	e_{01}	e_{02}	e_{03}	$1+e_{0\infty}$
e_1	$-e_{01}$	0	e_{12}	e_{13}	$e_{1\infty}$
e_2	$-e_{02}$	$-e_{12}$	0	e_{23}	$e_{2\infty}$
e_3	$-e_{03}$	$-e_{13}$	$-e_{23}$	0	$e_{3\infty}$
e_∞	$-1-e_{0\infty}$	$-e_{1\infty}$	$-e_{2\infty}$	$-e_{3\infty}$	0

$\wedge$	w_0	w_1	w_2	w_3	w_∞
w_0	0	w_{01}	w_{02}	w_{03}	$w_{0\infty}$
w_1	$-w_{01}$	0	w_{12}	w_{13}	$w_{1\infty}$
w_2	$-w_{02}$	$-w_{12}$	0	w_{23}	$w_{2\infty}$
w_3	$-w_{03}$	$-w_{13}$	$-w_{23}$	0	$w_{3\infty}$
w_∞	$-w_{0\infty}$	$-w_{1\infty}$	$-w_{2\infty}$	$-w_{3\infty}$	0

A *circle* passing through three points a, b, and c is $P_a \wedge P_b \wedge P_c$. A *sphere* passing through four points a, b, c, and d is $P_a \wedge P_b \wedge P_c \wedge P_d$. A *line* passing through two points a and b is $P_a \wedge P_b \wedge w_\infty$. A *plane* passing through three points a, b, and c is $P_a \wedge P_b \wedge P_c \wedge w_\infty$. A *translator* is defined by $\exp(-\frac{d}{2}w_\infty)P\exp(\frac{d}{2}w_\infty)$ where $d = \alpha w_1 + \beta w_2 + \gamma w_3$, $(\alpha, \beta, \gamma \in \mathbb{R})$. A *rotor* is defined by $\exp(-\frac{\theta}{2}B)P\exp(\frac{\theta}{2}B)$, where $\theta \in \mathbb{R}$ is an angle of rotation, $B = b_1 \wedge b_2$, and $b_i = \alpha_i w_1 + \beta_i w_2 + \gamma_i w_3$, $(\alpha_i, \beta_i, \gamma_i \in \mathbb{R}, i = 1, 2)$. A *dilator* is $\exp(-\frac{\lambda}{2}w_0 \wedge w_\infty)P\exp(\frac{\lambda}{2}w_0 \wedge w_\infty)$, where $\lambda \in \mathbb{R}$ is a scaling factor.

Example 1. *In the following, we show some simple computations of elements in* W*:*

$$\begin{aligned}
w_{01} = w_0 \wedge w_1 &= -w_1 \wedge w_0, \\
w_\infty * w_0 &= -1 - w_{0\infty}, \\
w_{0\infty} * w_{123} &= -w_{023\infty}, \\
w_{12} * w_{3\infty} &= w_{123\infty}, \\
w_{3\infty} * w_{3\infty} &= (w_{3\infty})^2 = 0, \\
w_{0\infty} * w_{0\infty} &= (w_{0\infty})^2 = 0, \\
w_{12} * w_{12} &= (w_{12})^2 = -1, \\
w_{12} &= e_{12}, \text{ and} \\
w_{01\infty} &= e_{01\infty} - e_1.
\end{aligned}$$

The following examples are computations of the exp *function. Since* $(w_{3\infty})^2 = 0$, $(w_{12})^2 = -1$ *and* $(w_{0\infty})^2 = 1$, *we have*

$$\begin{aligned}
\exp(\frac{w_3}{2}w_\infty) &= 1 + \frac{1}{2}w_{3\infty} + \frac{1}{2!}(\frac{1}{2}w_{3\infty})^2 + \cdots \\
&= 1 + \frac{1}{2}w_{3\infty}, \\
\exp(\frac{\theta}{2}w_{12}) &= 1 + \frac{\theta}{2}w_{12} + \frac{1}{2!}(\frac{\theta}{2}w_{12})^2 + \frac{1}{3!}(\frac{\theta}{2}w_{12})^3 + \cdots \\
&= \{1 - \frac{1}{2!}(\frac{\theta}{2})^2 + \cdots\} + \{\frac{\theta}{2} - \frac{1}{3!}(\frac{\theta}{2})^3 + \cdots\}w_{12} \\
&= \cos\frac{\theta}{2} + w_{12}\sin\frac{\theta}{2}, \text{ and} \\
\exp(\frac{\lambda}{2}(1 + w_\infty w_0)) &= \exp(-\frac{\lambda}{2}w_{0\infty}) \\
&= 1 + \frac{\lambda}{2}w_{0\infty} + \frac{1}{2!}(\frac{\lambda}{2}w_{0\infty})^2 + \frac{1}{3!}(\frac{\lambda}{2}w_{0\infty})^3 + \cdots
\end{aligned}$$

$$\begin{aligned} &= \{1+\frac{1}{2!}(\frac{\lambda}{2})^2+\cdots\}+\{\frac{\lambda}{2}+\frac{1}{3!}(\frac{\lambda}{2})^3+\cdots\}w_{0\infty} \\ &= \cosh\frac{\lambda}{2}-w_{0\infty}\sinh\frac{\lambda}{2}. \end{aligned}$$

3 Mathematica module for CGA

The proposed *Mathematica* module has various functions. For example, *CGAProduct*, *OuterProduct* and *InnerProduct* are functions for computing products ($*$, $\wedge$ and $\cdot$) of two CGA elements. Moreover, *Pnt, Cir, Trs*, etc. are functions for creating a CGA element which represents a point, a circle, a translator, etc. The figure correspoinding to a CGA element X is a subset $Fig(X) := \{(x,y,z) \in \mathbb{R}^3 \,|\, X \wedge P_{(x,y,z)} = 0\}$ of $\mathbb{R}^3$. We implemented a function *CGAOutput3D* for drawing the corresponding figure in $\mathbb{R}^3$ for a given CGA element. *CGAEquationCheck* is a function for checking the equality of figures represented by two CGA elements. Let $X = \sum_{S\subset\mathcal{A}} a_S w_S \quad (a_S \in \mathbb{R})$. Then we have $X \wedge P_{(x,y,z)} = \sum_{S\subset\mathcal{A}} p_S w_S$, where p_S are polynomials in $\mathbb{R}[x,y,z]$ and $Fig(X) = \{(x,y,z) \,|\, p_S = 0\,(S \subset \mathcal{A})\}$. We list complete equations for $Fig(X)$ in Appendix. We compute a *Gröbner basis* of those equations. We note that there remains at most one degree 2 equation in the Gröbner basis for this case. So we can detect the dimension of a figure by the number of elements in the Gröbner basis. That is the dimension is 3 minus the number of elements. If the number is three, then the figure is a finite set of points. If the number is two, then we can draw the figure using *ParametricPlot3D* function with one variable. If the number is one, then we use *ContourPlot3D* function with two variables. Our *CGAOutput3D* function automatically choose an appropriate draw function in *Mathematica*.

We show two examples. First, we consider the intersection of a sphere and a plane. Let S_1 be the sphere passing through four points $(3,0,0)$, $(0,3,1)$, $(0,2,3)$, and $(-3,3,2)$, H_1 the plane which passes through three points $(3,0,1)$, $(0,3,1)$, and $(0,2,1)$. That is, $S_1 = P_{(3,0,0)} \wedge P_{(0,3,1)} \wedge P_{(0,2,3)} \wedge P_{(-3,3,2)} = 3w_{012\infty} - 33w_{013\infty} + 37w_{023\infty} + 192w_{123\infty} - 18w_{0123}$ and $H_1 = P_{(3,0,1)} \wedge P_{(0,3,1)} \wedge P_{(0,2,1)} \wedge w_\infty = 3w_{012\infty} + 3w_{123\infty}$. The intersection of S_1 and H_1 is a CGA element $S_1 \cdot H_1^* = -99w_{01\infty} + 111w_{02\infty} + 567w_{12\infty} + 99w_{13\infty} - 111w_{23\infty} + 54w_{012} + 54w_{123}$, where H_1^* is the dual of H_1 i.e. $H_1^* = H_1 * (-w_{0123\infty})$ (cf. [12]). Figure 3 represents the intersection of plane H_1 and sphere S_1.

$$\begin{aligned} Fig(S_1\cdot H_1^*) &= \{(x,y,z)\in\mathbb{R}^3 \,|\, -1+z=0, -180+37x+9x^2+33y+9y^2=0\} \\ &= \{(x,-\frac{11}{6}\pm\frac{1}{6}\sqrt{841-148x-36x^2},1)\in\mathbb{R}^3 \,|\, x\in\mathbb{R}\wedge 841-148x-36x^2>0\}. \end{aligned}$$

Since the number of elements in the Gröbner basis is two, an element in the set is expressed with one variable. In this example, we can draw a circle using the *Mathematica* function $ParametricPlot3D$.

Next, we consider the intersection of a circle and a plane. Let C_1 be the circle which passes through three points $(3,0,2)$, $(0,3,1)$, and $(0,3,0)$, that is $C_1 = P_{(3,0,2)} \wedge P_{(0,3,1)} \wedge P_{(0,3,0)} = \frac{3}{2}w_{01\infty} - \frac{3}{2}w_{02\infty} - w_{03\infty} - \frac{9}{2}w_{12\infty} + \frac{27}{2}w_{13\infty} - \frac{33}{2}w_{23\infty} + 3w_{013} - 3w_{023} - 9w_{123}$. The intersection of plane H_1 and circle C_1 is a CGA element $C_1 \cdot H_1^* = -3w_{0\infty} + 45w_{1\infty} - 54w_{2\infty} - 3w_{3\infty} - 9w_{01} + 9w_{02} + 27w_{12} + 9w_{13} - 9w_{23}$.

$$\begin{aligned} Fig(C_1\cdot H_1^*) &= \{(x,y,z)\in\mathbb{R}^3 \,|\, -1+z=0, -3-8y+3y^2=0, -3+x+y=0\} \\ &= \{(\frac{5+\sqrt{7}}{3},\frac{4-\sqrt{7}}{3},1),(\frac{5-\sqrt{7}}{3},\frac{4+\sqrt{7}}{3},1)\}. \end{aligned}$$

Since the number of elements in the Gröbner basis is three, we can completely solve the equations. Therefore, we draw these points using the *Mathematica* function *ListPointPlot*3*D*. Figure 4 represents the intersection of plane H_1 and circle C_1.

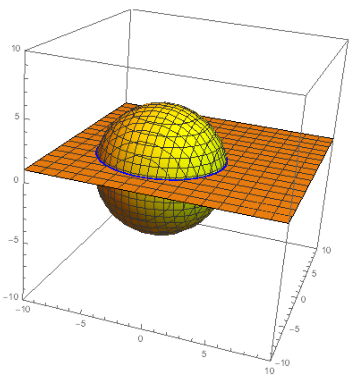

Figure 3: Sphere and Plane

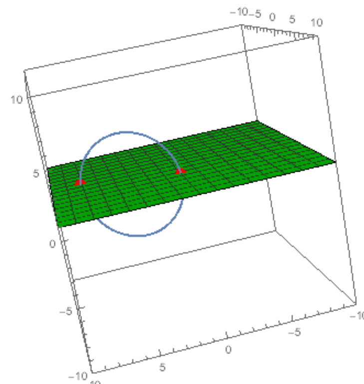

Figure 4: Plane and Circle

We also implemented a function to check whether the figures of two CGA elements are equal. Let $X_1 = w_{0\infty} - 6w_{1\infty} - 5w_{2\infty} - 4w_{3\infty} + w_{01\infty} + w_{02\infty} + w_{03\infty} + 6w_{12\infty} + 5w_{13\infty} + 6w_{23\infty} + w_{012\infty} + w_{013\infty} + w_{023\infty} - 5w_{123\infty} + w_{0123\infty} + w_{01} + w_{02} + w_{03} - w_{12} - 2w_{13} - w_{23} + w_{012} + w_{013} + w_{023} + 2w_{123} + w_{0123}$ and $X_2 = w_0 + w_1 + 2w_2 + 3w_3 + 7w_\infty$. The appearances of X_1 and X_2 are different, but the figures of X_1 and X_2 are same and it is $\{(1, 2, 3)\}$. Our implemented function *CGAEquationCheck* can check the equality of figures for any two CGA elements using Gröbner bases and the *PolynomialMod* function in *Mathematica*.

4 Two-dimensional origami folding

We follow the formulations introduced by Ida et al. [6, 13, 5, 2]. First, we use a data structure called **origami graph** $O = (\Pi, \sim, \succ)$, where Π is a set of faces, $\sim$ is an adjacency relation of faces, and $\succ$ is a superposition relation (cf. [13, 5]).

Our *Mathematica* module has various functions. *Ori* is the function that folds origami and automatically updates the origami graph O. Arguments of $\mathrm{Ori}[O, m, F, \theta]$ are a current origami graph $O = (\Pi, \sim, \succ)$, a fold line m, a faces set F for folding and an angle of rotation $\theta \in \{\pi, -\pi, 0\}$, where F may be a part of faces for folding and θ indicates a valley fold or a mountain fold. We use $\theta = 0$ for an operation which does not fold but just divide faces using a given fold line and a face set. To determine a folding line from given points and lines, we follow the Huzita-Hattori axioms and we implemented Mathematica functions *Ori1* to *Ori7*. We recall the Huzita-Hatori axioms using our functions as follows[2]:

Axiom 1: Given two points p_1 and p_2, there is a **unique fold line** ($\mathrm{Ori1}[p_1, p_2]$) that passes through both points (Figure 5).

Axiom 2: Given two points p_1 and p_2, there is a **unique fold line** ($\mathrm{Ori2}[p_1, p_2]$) that places p_1 onto p_2 (Figure 6).

Axiom 3: Given two lines L_1 and L_2, there are at most **two fold lines** ($\mathrm{Ori3}[L_1, L_2, flag]$) that place L_1 onto L_2. There are at most two choices of folding line. We use "flag" to indicate a folding line(Figure 7).

Axiom 4: Given a point p_1 and a line L_1, there is a **unique fold line** (Ori4$[p_1, L_1]$) that passes through point p_1 and is perpendicular to L_1 (Figure 8).

Axiom 5: Given two points p_1 and p_2 and a line L_1, there are at most **two fold lines** (Ori5$[p_1, p_2, L_1, flag]$) that place p_2 onto L_1 and pass through p_1. There are at most two choices of folding line. We use "flag" to indicate a folding line (Figure 9).

Axiom 6: Given two points p_1 and p_2 and two lines L_1 and L_2, there are at most **three fold lines** (Ori6$[p_1, L_1, p_2, L_2, flag]$) that place p_1 onto L_1 and p_2 onto L_2. There are at most three choices of folding line. We use "flag" to indicate a folding line (Figure 10).

Axiom 7: Given one point p and two lines L_1 and L_2, there is a **unique fold line** (Ori7$[p_1, L_1, L_2]$) that places p onto L_2 and is perpendicular to L_1 (Figure 11).

Figure 5: Ori1$[p_1, p_2]$

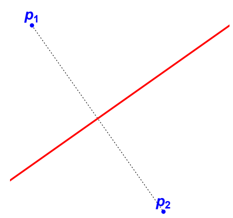

Figure 6: Ori2$[p_2, p_2]$

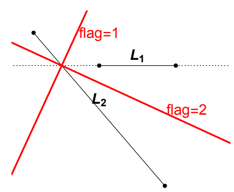

Figure 7: Ori3$[L_1, L_2, flag]$

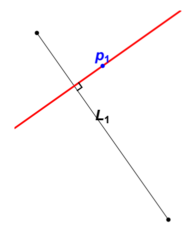

Figure 8: Ori4$[p_1, L_1]$

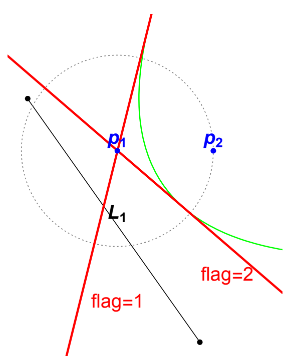

Figure 9: Ori5$[p_1, p_2, L_1, flag]$

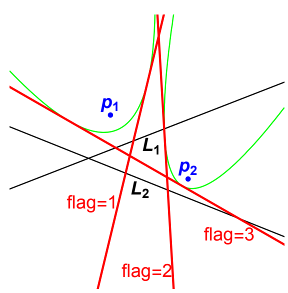

Figure 10: Ori6$[p_1, L_1, p_2, L_2, flag]$

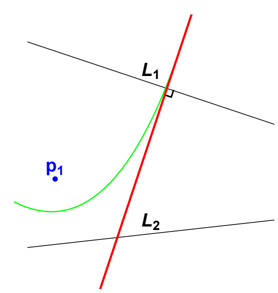

Figure 11: Ori7$[p_1, L_1, L_2]$

Example 2. *We can make any origami by successive applications of origami functions. The following is a sequence of origami functions for making Kabuto. Intermediate results are displayed in Figures 12 to 15.*

We first define coordinates of vertices (P_1, P_2, P_3, P_4) of Origami.

$$
\begin{aligned}
& && P_1 = \{0,0\};\; P_2 = \{0,10\};\; P_3 = \{10,10\};\; P_4 = \{10,0\}; \\
O_0 &=& &\text{FirstO}; && \cdots (\mathit{Figure}\ 12). \\
O_1 &=& &\text{Ori}[O_0, \text{Ori2}[P_3, P_1], \{1\}, \pi]; \\
O_2 &=& &\text{Ori}[O_1, \text{Ori2}[P_4, P_3], \{3\}, \pi]; \\
O_3 &=& &\text{Ori}[O_2, \text{Ori2}[P_1, P_4], \{5\}, -\pi];
\end{aligned}
$$

In this example, we explicitly define points P_5, P_6, and P_7, but these points are automatically defined after folding operations as the coordinate of a vertex of some folded face.

$$
\begin{aligned}
& && P_5 = \frac{P_1 + P_3}{2};\; P_6 = \frac{P_1 + P_2}{2};\; P_7 = \frac{P_1 + P_4}{2}; \\
O_4 &=& &\text{Ori}[O_3, \text{Ori2}[P_5, P_1], \{4, 10\}, \pi]; && \cdots (\mathit{Figure}\ 13). \\
O_5 &=& &\text{Ori}[O_4, \text{Ori2}[P_6, P_5], \{13\}, 0]; \\
O_6 &=& &\text{Ori}[O_5, \text{Ori2}[P_7, P_5], \{29\}, 0]; \\
& && P_8 = \frac{P_1 + P_5}{2};\; P_9 = \frac{P_5 + P_6}{2};\; P_{10} = \frac{P_5 + P_7}{2}; \\
O_7 &=& &\text{Ori}[O_6, \text{Ori3}[\text{p2line}[P_8, P_5], \text{p2line}[P_9, P_8], 2], \{27\}, \pi]; \\
O_8 &=& &\text{Ori}[O_7, \text{Ori3}[\text{p2line}[P_8, P_5], \text{p2line}[P_{10}, P_8], 2], \{59\}, \pi];
\end{aligned}
$$

The P_{11} is a user defined point. In this example, we define P_{11} as a midpoint of P_8 and P_5.

$$
\begin{aligned}
& && P_{11} = \frac{P_8 + P_5}{2}; \\
O_9 &=& &\text{Ori}[O_8, \text{Ori2}[P_{11}, P_1], \{15\}, \pi]; && \cdots (\mathit{Figure}\ 14). \\
O_{10} &=& &\text{Ori}[O_9, \text{Ori1}[P_7, P_6], \{31\}, \pi]; \\
O_{11} &=& &\text{Ori}[O_{10}, \text{Ori2}[P_7, P_6], \{11\}, -\pi]; && \cdots (\mathit{Figure}\ 15).
\end{aligned}
$$

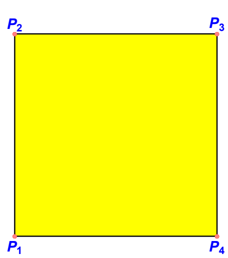

Figure 12: GOutput2D(O_0)

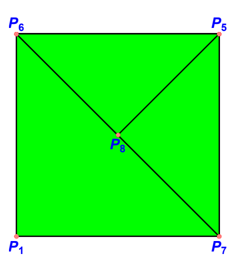

Figure 13: GOutput2D(O_4)

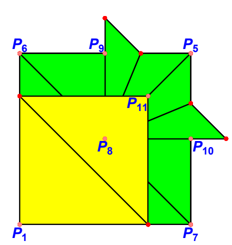

Figure 14: GOutput2D(O_9)

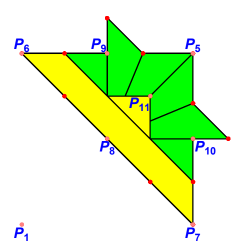

Figure 15: GOutput2D(O_{11})

GOutput2D is the function to draw an origami graph O, as viewed from the top and in a tab layer display in a 2D space (Figure 16). *GOutput3D* is the function to draw an origami graph

O by layer display in a 3D space (Figure 17). *OriBack* is the function to open folded origami, that is , the converse operation of *Ori*. This function can create a development view.

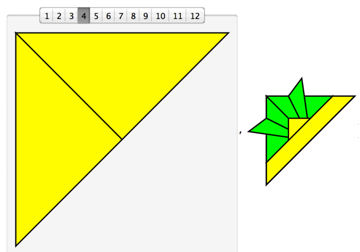

Figure 16: GOutput2D

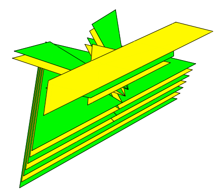

Figure 17: GOutput3D

5 Three-dimensional origami folding

When we transform a point x to a point y, there are various types of representations that can be used to express a transformation. We often use a vector-matrix representation. We express operation M as a 3×3 matrix and point x, y as a 3×1 vector. A transform is represented as an equation $Mx = y$. This representation requires the definition of two data structures (matrix and vector). We should calculate the act operators and product operator between two data types. Therefore, this is sometimes difficult to verify formally. However, CGA representation requires defining only one data structure (a set of CGA elements). In CGA representation, we can express both operation R and point x, y as a CGA element. Moreover, the transformation of point x to point y is represented as an algebraic equation $R{*}x{*}R^{-1} = y$ ($R{*}R^{-1} = R^{-1}{*}R = 1$). Therefore, all calculations can be achieved simply by symbolic computation of the product operation $*$, making the operation easy to formalize and verify. This is the reason why we use our CGA representation.

We express the method by which to calculate a CGA operation R from given fold operation as fold line m and angle $\theta \in \mathbb{R}$. We consider fold line m, which is a directed line, as a pair of vectors (v_1, v_2), and the direction is given by the vector $v = \frac{v_2 - v_1}{\|v_2 - v_1\|}$. A fold operation can be represented as the rotor on fold line m. As such, we define the rotor around the origin Rot and translator T as follows:

$$\begin{aligned} Rot &= \cos\frac{\theta}{2} - B\sin\frac{\theta}{2} \\ & \quad \text{(bivector } B = v * (-w_{123}) \text{ which is the plane of the rotation)}, \\ T &= 1 - \frac{v_1}{2} w_\infty . \end{aligned}$$

Since Rot is a rotor around the origin, we should rewrite the equation as a rotor around v_1. This can be achieved by using a translation defined by T and T^{-1}. We define $R = T{*}Rot{*}T^{-1}$, where R corresponds to the fold operation. This is the rotor of the left-hand direction on fold line m.

Example 3. *We present an example of the calculation of CGA operation R. Let the fold operation be a $\frac{\pi}{3}$ folding on fold line $m = (v_1, v_2)$, $(v_1 = 6w_1, v_2 = 4w_1 + 10w_2)$ (Figure 18).*

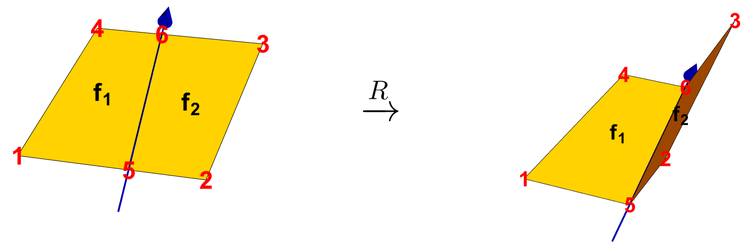

Figure 18: $\frac{\pi}{3}$ folding

$$
\begin{aligned}
v &= \frac{4w_1+10w_2-6w_1}{\|4w_1+10w_2-6w_1\|} = -\frac{w_1}{\sqrt{26}}+\frac{5w_2}{\sqrt{26}} \\
B &= v*(-w_{123}) = \frac{5w_{13}}{\sqrt{26}}+\frac{w_{23}}{\sqrt{26}} \\
Rot &= \cos\frac{\pi}{6}-B\sin\frac{\pi}{6} = \frac{\sqrt{3}}{2}-\frac{5w_{13}}{2\sqrt{26}}-\frac{w_{23}}{2\sqrt{26}} \\
T &= 1-3w_{1\infty},\ T^{-1} = 1+3w_{1\infty} \\
R &= T*Rot*T^{-1} = \frac{\sqrt{3}}{2}-\frac{5w_{13}}{2\sqrt{26}}-\frac{w_{23}}{2\sqrt{26}}+\frac{15w_{3\infty}}{\sqrt{26}}
\end{aligned}
$$

6 Simple proof using CGA equations

Various figures can be obtained by folding an origami. We can prove their geometric properties by calculating CGA equations. For example, when we fold a quadratic prism from rectangular origami (Figure 19), we show that points P_1 and P_2 are in the same position.

Let P_1, P_2, P_3, and P_4, which are the vertices of the rectangular origami, be the CGA points $P_1 = P_{(0,0,0)} = w_0$, $P_2 = P_{(A,0,0)} = w_0 + Aw_1 + \frac{A^2}{2}w_\infty$, $P_3 = P_{(A,B,0)} = w_0 + Aw_1 + Bw_2 + \frac{A^2+B^2}{2}w_\infty$ and $P_4 = P_{(0,B,0)} = w_0 + Bw_2 + \frac{B^2}{2}w_\infty$ $(A > 0 \wedge B > 0)$. We can express fold operations R_1, R_2, and R_3 in CGA and the CGA operation R as a combination of R_1, R_2, and R_3, as follows: $R_1 = \frac{1}{\sqrt{2}} + \frac{w_{13}}{\sqrt{2}} - \frac{A}{4\sqrt{2}}w_{3\infty}$, $R_2 = \frac{1}{\sqrt{2}} + \frac{w_{13}}{\sqrt{2}} - \frac{A}{2\sqrt{2}}w_{3\infty}$, $R_3 = \frac{1}{\sqrt{2}} + \frac{w_{13}}{\sqrt{2}} - \frac{3A}{4\sqrt{2}}w_{3\infty}$, $R = R_3 * R_2 * R_1 = \frac{1}{\sqrt{2}} + \frac{w_{13}}{\sqrt{2}} + \frac{A}{2\sqrt{2}}w_{1,\infty} - \frac{A}{2\sqrt{2}}w_{3\infty}$.

We can calculate the CGA equation using the function in our *Mathematica* module

$$R * P_1 * R^{-1} = w_0 + Aw_1 + \frac{A^2}{2}w_\infty = P_2.$$

This equation confirms that points P_1 and P_2 are located at the same position.

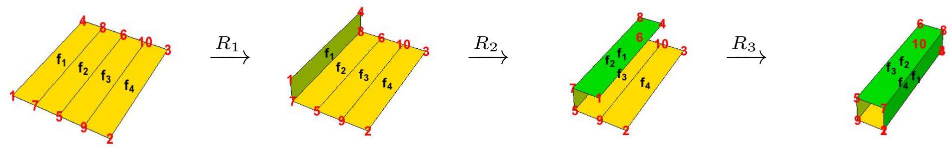

Figure 19: Folding the quadratic prism

7 Conclusion

We implemented functions that realize CGA operations using symbolic computations in *Mathematica* [1]. Our module includes functions for the geometric product, the inner product, and the outer product of a CGA which is an extension of $\mathbb{R}^3$ with e_0 and e_∞. We also implemented a drawing function for figures in $\mathbb{R}^3$ which correspond to CGA elements. Furthermore, we implemented a function to check whether two figures of CGA elements are equal. Next, we considered an application of our CGA module to 3D origami folding. Following the formulation of 2D origami folding introduced by Ida et al., we extended the folding function. Using our module, we can study 3D origami folding, in particular the properties of a sequence of folding procedures. Finally, we presented a simple proof of the 3D origami property using a symbolic computation of the CGA equations. Future research include a precise implementation of 3D origami folding functions. We intend to implement a function for judging the collision of faces in 3D origami folding. Hopefully, we will be able to introduce various useful 3D origami motions described by CGA operations.

Acknowledgement

This work was partially supported by the Core Research for Evolutional Science and Technology (CREST) Program titled "Mathematics for Expressive Image Synthesis" of the Japan Science and Technology Agency (JST).

References

[1] P. Colapinto. Versor spatial computing with conformal geometric algebra. Master's thesis, University of California Santa Barbara, 2011.

[2] F. Ghourabi, A. Kasem, and C. Kaliszyk. Algebraic analysis of huzita's origami operations and their extensions. In *Automated Deduction in Geometry*, volume 7993 of *Lecture Notes in Computer Science*, pages 143–160. Springer, 2013.

[3] T. Ida. Eos project. http://www.i-eos.org:8080/ieos.

[4] T. Ida. Huzita's basic origami fold in geometric algebra. In *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2014)*, pages 11–13. IEEE computer society, 2014.

[5] T. Ida and H. Takahashi. Origami fold as algebraic rewriting. *Journal of Symbolic Computation*, 45(4):393–413, 2010.

[1]cf. https://github.com/KyushuUniversityMathematics/MathematicaCGA

[6] T. Ida, H. Takahashi, M. Marin, A. Kasem, and F. Ghourabi. Computational origami system eos. In *Proc. 4th International Conference of Origami Science, Mathematics and Education (4OSME)*, pages 285–293, 2009.

[7] L. Kavan, S. Collins, J. Zara, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transaction on Graphics*, 27(4):105:1–105:23, 2008.

[8] M. Kondo and T. Matsuo. A mathematica module for conformal geometric algebra. In *Symposium on Mathematical Progress in Expressive Image Synthesis (MEIS2014)*, volume 58 of *MI Lecture Note Series*, pages 132 – 136. Kyushu University, 2014.

[9] A. Macdonald. A survey of geometric algebra and geometric calculus, 2014.

[10] G. Matsuda, S. Kaji, and H. Ochiai. Anti-commutative dual complex numbers and 2D rigid transofmation. In *Symposium on Mathematical Progress in Expressive Image Synthesis (MEIS2013)*, MI Lecture Note Series, pages 128 – 133. Kyushu University, 2013.

[11] G. Papagiannakis. Geometric algebra rotors for skinned character animation blending. In *SIGGRAPH Asia 2013 Technical Briefs*, pages 11:1–11:6. ACM, 2013.

[12] C. Perwass. *Geometric Algebra with Applications in Engineering*, volume 4 of *Geometry and Computing*. Springer, 2009.

[13] H. Takahashi. *A Study on algorithms for computational origami(Japanese)*. PhD thesis, University of Tsukuba, 2009.

Appendix

Let $X = \sum_{S \subset \mathcal{A}} a_S w_S$, $(a_S \in \mathbb{R})$. Equations for $X \wedge P_{(x,y,z)} = 0$ are listed as follows:

$$
\begin{aligned}
& X \wedge P_{(x,y,z)} = 0 \\
\Leftrightarrow & \begin{cases}
a_\phi & = 0 \\
a_\phi x & = 0 \\
a_\phi y & = 0 \\
a_\phi z & = 0 \\
a_\phi \frac{x^2+y^2+z^2}{2} & = 0 \\
a_{\{0\}} x - a_{\{1\}} & = 0 \\
a_{\{0\}} y - a_{\{2\}} & = 0 \\
a_{\{0\}} z - a_{\{3\}} & = 0 \\
a_{\{0\}} \frac{x^2+y^2+z^2}{2} - a_{\{\infty\}} & = 0 \\
a_{\{1\}} y - a_{\{2\}} x & = 0 \\
a_{\{1\}} z - a_{\{3\}} x & = 0 \\
a_{\{1\}} \frac{x^2+y^2+z^2}{2} - a_{\{\infty\}} x & = 0 \\
a_{\{2\}} z - a_{\{3\}} y & = 0 \\
a_{\{2\}} \frac{x^2+y^2+z^2}{2} - a_{\{\infty\}} y & = 0 \\
a_{\{3\}} \frac{x^2+y^2+z^2}{2} - a_{\{\infty\}} z & = 0 \\
-a_{\{0,2\}} x + a_{\{0,1\}} y + a_{\{1,2\}} & = 0 \\
-a_{\{0,3\}} x + a_{\{0,1\}} z + a_{\{1,3\}} & = 0 \\
a_{\{0,1\}} \frac{x^2+y^2+z^2}{2} - a_{\{0,\infty\}} x + a_{\{1,\infty\}} & = 0 \\
-a_{\{0,3\}} y + a_{\{0,2\}} z + a_{\{2,3\}} & = 0 \\
a_{\{0,2\}} \frac{x^2+y^2+z^2}{2} - a_{\{0,\infty\}} y + a_{\{2,\infty\}} & = 0 \\
a_{\{0,3\}} \frac{x^2+y^2+z^2}{2} - a_{\{0,\infty\}} z + a_{\{3,\infty\}} & = 0 \\
a_{\{2,3\}} x - a_{\{1,3\}} y + a_{\{1,2\}} z & = 0 \\
a_{\{1,2\}} \frac{x^2+y^2+z^2}{2} + a_{\{2,\infty\}} x - a_{\{1,\infty\}} y & = 0 \\
a_{\{1,3\}} \frac{x^2+y^2+z^2}{2} + a_{\{3,\infty\}} x - a_{\{1,\infty\}} z & = 0 \\
a_{\{2,3\}} \frac{x^2+y^2+z^2}{2} + a_{\{3,\infty\}} y - a_{\{2,\infty\}} z & = 0 \\
a_{\{0,2,3\}} x - a_{\{0,1,3\}} y + a_{\{0,1,2\}} z - a_{\{1,2,3\}} & = 0 \\
a_{\{0,1,2\}} \frac{x^2+y^2+z^2}{2} + a_{\{0,2,\infty\}} x - a_{\{0,1,\infty\}} y - a_{\{1,2,\infty\}} & = 0 \\
a_{\{0,1,3\}} \frac{x^2+y^2+z^2}{2} + a_{\{0,3,\infty\}} x - a_{\{0,1,\infty\}} z - a_{\{1,3,\infty\}} & = 0 \\
a_{\{0,2,3\}} \frac{x^2+y^2+z^2}{2} + a_{\{0,3,\infty\}} y - a_{\{0,2,\infty\}} z - a_{\{2,3,\infty\}} & = 0 \\
a_{\{1,2,3\}} \frac{x^2+y^2+z^2}{2} - a_{\{2,3,\infty\}} x + a_{\{1,3,\infty\}} y - a_{\{1,2,\infty\}} z & = 0 \\
a_{\{0,1,2,3\}} \frac{x^2+y^2+z^2}{2} - a_{\{0,2,3,\infty\}} x + a_{\{0,1,3,\infty\}} y - a_{\{0,1,2,\infty\}} z + a_{\{1,2,3,\infty\}} & = 0.
\end{cases}
\end{aligned}
$$