折紙の展開図専用エディタ（ORIPA）の開発および展開図からの折りたたみ形状推定

三谷 純[†,††]

近年では折紙作品の創作に「設計」の概念が導入されるようになり，複雑な作品を効率的に創作できるようになっているが，それには「展開図」に対する考察が必要不可欠である．また，計算機の普及とともに，折紙の研究に計算機を用いることも一般的になりつつある．そこで，折紙の基本データである展開図を計算機に効率的に入力するための専用エディタを開発した．このエディタには，折紙の展開図に見られる特徴的な折り線の構造を容易に入力できる機能を組み込んだ．さらに，入力された展開図から折りたたみ後の形状を推定する手法を考案し実装を行った．この手法では，折り線または紙の輪郭線によって囲まれる閉領域を折紙の構成要素とし，折りたたみ後にそれぞれの要素がどの位置に移動し，重なり順がどのようになるかを計算することで，折りたたみ後の形状を推定する．これにより，折紙の展開図および折りたたみ後のデジタルデータを容易に取得することが可能となり，今後の計算機を用いた折紙の研究のための基盤ツールとして活用されることが期待できる．

Development of Origami Pattern Editor (ORIPA) and a Method for Estimating a Folded Configuration of Origami from the Crease Pattern

Jun Mitani[†,††]

In these years, the concept of 'design' has been introduced to the stage of creation of Origami works and it has become possible to create complicated Origami works effectively. For that, an examination on a crease pattern is indispensable. It has been becoming popular to use computers to study about Origami together with the spread of them. We developed a special editor for inputting the crease pattern of Origami to the computer efficiently. Some special features which make easier to input characteristic line segments were incorporated into the editor, the feature to calculate the folded configuration from a crease pattern was also introduced. In this method, the closed space surrounded by the folded lines is considered as a component. It calculates the folded configuration by estimating how each component moves and in what order each line overlaps. With this tool, it becomes easy to build Origami digital data of a crease pattern and the folded configuration and it can be used as one of the fundamental tools for studying Origami using computers.

1. はじめに

紙を折ることで形を表現する「折紙」は日本に古くから伝わる文化の１つであり，教育の場での活用や趣味の１つとして幅広い世代に親しまれている．近年では世界的にも広く認知され，学術的な研究も多くされている．特に最近開催された折紙に関する国際会議，4OSME（The Fourth International Conference on Origami in Science, Mathematics and Education）では数学，工学，建築，情報，生物，教育，芸術，文化にわたる広範囲の研究者が集い，それぞれの分野における「ものを折りたたむこと」に関する研究成果が報告された．

折紙作品の創作においては，近年になって「設計」の概念が持ち込まれるようになり，従来の試行錯誤に基づく創作活動から得られるよりも，より複雑で精緻な作品が数多く生み出されるようになっている．折紙には，正方形の紙に対して切り込みを入れずに，折り操作だけで目的の形を作らなければならないという厳しい制約がある．このため，意図した形を作り出すためには，目的とする形状の各部位を展開図上にどのように配置するかをあらかじめ考慮する必要があり，複雑な作品の設計においては「展開図」に対する考察がなくてはならないものになっている．新しく創作された折紙の折り方を他者に伝える方法には「折り図

† 筑波大学大学院システム情報工学研究科
Department of Computer Science, University of Tsukuba
†† 科学技術振興機構さきがけ
PRESTO, JST

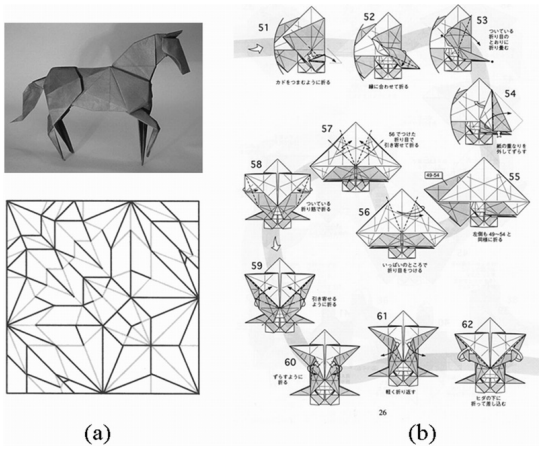

図 1 馬[1] の (a) 展開図と (b) 折り図の一部
Fig. 1 (a) Crease pattern and (b) diagram (a part) of a horse [1].

(**図 1** (b))」と呼ばれる，折り工程を図で表現したものが一般的に用いられるが，複雑な作品の折り図の作成は労力を要する作業であり，このことが様々な作品を伝達することを困難にしている．そのため，折紙創作家の間では複雑な作品を伝達する手段として展開図を用いることが広く行われている．このように，折紙の作品を扱ううえで，その「展開図（図 1 (a))」が重要な役割を果たすようになってきている．

近年では計算機の普及とともに，折紙の研究に計算機を用いることも一般的になりつつあるが，計算機内に折紙の形状データを構築するための手法はいまだ確立されていない．一般的な CG のインタフェースで 1 つ 1 つモデリングするのは手間がかかりすぎるため，ユーザと対話的に擬似的な紙を折る操作でモデルを構築する研究もされているが，実用に供されるには至っていない．そこで本研究では，折紙を折りたたんだ後の形ではなく，折紙の「展開図」を効率的に作図するための専用エディタ（ORIPA: ORIgami PAttern editor）の開発を行い，折りたたみ後の形状を入力された展開図から再現することを行った．これにより，折紙の形状を計算機に入力する手間を大幅に軽減することができる．このエディタには，折紙の展開図に特徴的に現れる線分の構造を効率的に入力するための機能を実装した．さらに，平坦に折りたたまれる作品に限り，入力された展開図が折紙の展開図として妥当であるか否かを判定し，妥当な展開図に対しては，折りたたみ後の形状を推定して表示する機能を実装した．この手法では，折り線または紙の輪郭線によって囲まれる閉領域を折紙の構成要素とし，折りたたみ後にそれぞれの要素がどの位置に移動し，重なり順がどのようになるかを計算することで，折りたたみ後の形状を推定する．これにより，折紙の展開図のデジタルデータを容易に構築することが可能となるため，このエディタが今後の折紙の研究の発展に寄与することが期待できる．

本稿の 2 章では関連する研究を紹介し，3 章で折紙専用エディタの展開図の作図機能について述べる．4 章では展開図から折りたたみ後の形状を推定する手法を述べる．5 章で考察と他のアプリケーションへの応用事例を紹介し，最後に 6 章でまとめと今後の展望を述べる．

2. 関 連 研 究

折紙は正方形の紙に対して折り操作を繰り返すことで様々な形を作るものであり，幾何の分野における研究題材として多く取り上げられている．これらの多くは折紙の展開図に対するものである[2),3)]．また，国内にとどまらず海外でも広く研究が行われており，近年では対象とする形状を折り出すための「設計」の概念が導入され，複雑な形状を持つ作品を効率的に作り出せるようになっている．文献 4) では具体的な折紙設計の手法が解説されている．折紙の設計においては，展開図上に主な折り線をどのように配置するか決定することが重要であり，展開図に対する深い考察が必要である．

近年の計算機の普及とともに折紙を計算機で扱う研究も多くされている．内田ら[5)] は紙の物理的な制約条件をもとに，折紙の展開図を構成する幾何学的要素から折りたたみ方法を推論するプログラムを提案した．この研究は本稿の後半で述べる展開図から折りたたみ形状を推定するテーマと非常に近いが，対象とする展開図を構成する折り線が「山折線」と「谷折線」だけでなく,「山折に折って開いた線」と「谷折に折って開いた線」も含まれる点が本稿で対象とする展開図と大きく異なる．つまり内田らが対象とした展開図には，最終的な形を決定するのに用いない折り線も含まれ，本稿が対象とする展開図よりも情報が多い．**図 2** は内田らが例題として用いた鶴の展開図であり，本稿で扱う鶴の展開図（図 5 (a)）と比較すると違いが明らかである．一般に折紙創作家らが作品の展開図として公開しているものには，折り操作の途中で副次的に発生する折り線は含まれず，折紙の機関誌「折紙探偵団」で紹介されている「展開図折り」（展開図から作品を折ること）が対象とする展開図にもこれらは含まれていない．そこで本稿でも，折りたたみ後の形状に直接寄与する折り線だけを含む展開図を対象とする．

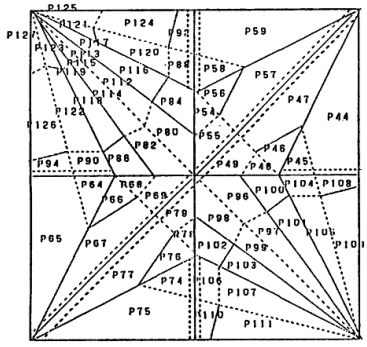

図 2 4 種類の折り線を持つ鶴の展開図[5)]
Fig. 2 Crease pattern of a crane with four types of line [5)].

Miyazaki ら[6)] は，計算機を用いて折紙をディスプレイに表示しながら対話的に操作する手法を提案した．紙を折る操作によって折紙の形状が逐次変化する際の，データ更新のアルゴリズムについての記述があり，そのプログラムは Web 上で公開されている[7)]．これは PC 上で動かすことができるが，入力すべき操作が実際の折紙の折り方と異なること，および複数枚の紙が多数重なり合う箇所で特定の部位を選択して折る操作が難しいことから，複雑な作品を作りあげることは困難である．Kato ら[8)] は，「折り図」の画像を計算機で解析することで，折り操作を推定し，それをもとに計算機内の折紙モデルを更新する手法を提案した．この手法は図中に含まれる矢印や折り線の情報を活用している．本稿では折紙の展開図を効率的に入力するエディタを開発し，それによって得られる展開図から折りたたみ後の形状を推定することで，計算機内に折紙のモデルを構築する手法を提案する．島貫ら[9),10)] は，折り操作によって得られる妥当な面の重なり順の算出方法を提案しているが，これは特定の切断面に着目して推定しているため，局所的な領域の重なり順を求める方法である．本稿の 4 章で述べる面の重なり順の判定手法は，とりうる可能性をすべて調べ上げるため，大局的な面の重なり順を決定できる．

Illustrator や FreeHand などのドローソフト，および 2 次元 CAD ソフトを使用することで，折紙の展開図を計算機上に作図できるが，これら一般的なドローソフトには垂直二等分線や角の二等分線など，折紙の展開図に特徴的に現れるパターン（3.2 節で述べる）を入力するには，複数のコマンドを実行する必要があり効率的でない．この問題は，手書きスケッチ入力から拘束条件を満たす幾何要素を効率的に入力する河合ら[11)] の研究を応用することで解決できるかもしれないが，作図された展開図が折紙の展開図として妥当なものであるかを判断することはできない．本研究で提案するエディタでは，効率的な作図を可能とするとともに，展開図の妥当性をチェックし保証することが可能である．また，ファイルフォーマットに XML 形式を採用しているため，クローズドな市販製品のファイル形式と異なり，今後の折り紙研究の普及に貢献することが可能であると考えられる．

3. 折紙の展開図の入力

すでに述べてきたように，折紙の研究では展開図を扱うことが重要である．そこで，折紙の展開図に特化したエディタ（ORIPA: ORIgami PAttern editor）を開発した．本章では，このエディタについて述べる．

3.1 折紙の展開図の特徴

一般的な折紙の展開図および本稿で対象とする展開図は「山折線」,「谷折線」および紙の輪郭を表す「輪郭線」の 3 種類の線分の集合から構成される．折り線は紙を折ることで生成されるが，折紙で用いられる（任意の線で折るのではなく，展開図上の幾何情報を基準に用いて紙を折る）折り方のパターンは有限であり，Huzita[12)] と Hatori[13)] によって発見された Huzita-Hatori axiom と呼ばれる次の 7 通りがすべてであることが Lang によって証明されている[14)]．

(1) 2 本の直線 L_1 と L_2 があるとき，L_1 を L_2 に合わせる直線を折る．
(2) 2 つの点 P_1 と P_2 があるとき，P_1 を P_2 に合わせる直線を折る．
(3) 2 つの点 P_1 と P_2 があるとき，P_1 と P_2 の両方を通る直線を折る．
(4) 1 つの点 P と 1 本の直線 L があるとき，P を通って L に垂直な直線を折る．
(5) 2 つの点 P_1 と P_2，1 本の直線 L があるとき，P_1 を L の上に乗せ，P_2 を通る直線を折る．
(6) 2 つの点 P_1 と P_2，2 本の直線 L_1 と L_2 があるとき，P_1 を L_1 の上に乗せ，P_2 を L_2 の上に乗せる直線を折る．
(7) 1 つの点 P，2 本の直線 L_1 と L_2 があるとき，P を L_1 の上に乗せ，L_2 に垂直な直線を折る．

このうち，一般的な折紙で行われる折り方は (1)〜(4) である．(5)〜(7) は数学的な探求から得られた特殊な折り方であり，一般的に用いられることはあまりない．(1) で得られる折り線は L_1 と L_2 の角の二等分線であり，(2) は 2 点の垂直二等分線，(3) は 2 点を通る直線，(4) は 1 点を通る垂線である．

3.2 エディタ機能の実装

本稿で提案するエディタでは，折紙の展開図入力を効率的に行うことを可能とするために，次に示す 9 通

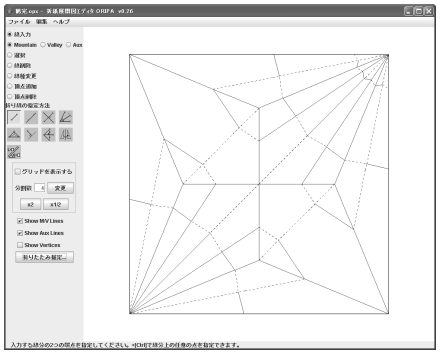

図 3 ORIPA の画面（鶴の基本形）
Fig. 3 Snap shot of ORIPA (Crane base).

りの方法で折り線の線分を入力する機能を実装した．
(1) 指定した角を 2 等分する線分
(2) 指定した 2 点の垂直二等分線
(3) 指定した 2 点を通り輪郭線まで伸びる線分
(4) 指定した点から指定した線分への垂線
(5) 指定した 3 点とその内心を結ぶ 3 本の線分
(6) 指定した線分と，線対称な位置にある線分
(7) 線分群の線対称コピー
(8) 指定した 2 点を端点に持つ線分
(9) 角度と長さを指定した線分

上記の (1)〜(4) は，Huzita-Hatori axiom によって生成される折り線を入力するための機能で，それぞれが前節であげた (1)〜(4) の折り方に対応している．(5)〜(7) は折紙の展開図に典型的に見られるパターンを効率的に入力するためのもので，(5) は鶴の展開図（**図 3**）の基本形や蛙の基本形（図 13）に現れる uniaxial bases と呼ばれる基本パターン[4] を，(6) は中割り折り（鶴の頭部の仕上げで用いられる折り方）によって生成されるパターンを効率的に入力することができる．(7) は対称な位置に存在する折り線の入力の手間を軽減するものである．(8)，(9) はその他の様々なパターンに汎用的に対応するための入力機能である．

なお，このエディタは PC 上に Java アプリケーションとして実装を行った（図 3）．作図した展開図のデータは他のアプリケーションでも活用できるよう XML 形式で出力できるようにし，Web 上での公開を行っている[15]．

一般的な CAD ソフトでも複数の手順を踏むことで上記と同等の機能を実現することはできるが，ここであげた折紙の展開図に特化した入力手法を備えることにより，効率的に展開図を入力することができる．

3.3 展開図の妥当性の判定

展開図はエディタで入力されるが，入力の時点では何ら制約が設けられていないため，入力された展開図が妥当な展開図でない可能性がある．ここでの「妥当でない展開図」とは，平坦に折りたたまれる折紙を想定して入力された展開図に対し，折り線のとおりに折り操作を行っても，実際には平坦に折りたたむことが不可能な展開図のことをいう．

与えられた展開図が平坦に折りたためるために必要十分な，展開図の内点に接続している折り線の条件（局所平坦条件）は文献 16)〜18) より明らかにされており，次のとおりである．
(1) 折り線の数は偶数である．
(2) 山折線の数と谷折線の数の差の絶対値は 2 である．
(3) 折り線のなす角の 1 つおきの和は 180° である．
(4) 折り線のなす角が鈍角のとき，その角をはさむ 2 つの線の折り線属性（山折/谷折）は等しい．

これらの条件をすべての内点が満たす展開図は平坦に折りたたむことができ，条件を満たさない展開図は平坦に折りたたむことができない．入力された展開図に対して，上記の条件を満たすか否かを調べることで，妥当でない展開図を識別できる．上記の判定は展開図の内点ごとに局所的に行うことができるため，本稿で提案するエディタ OIRPA では条件を満たさない部分を強調表示してユーザに示すことで，展開図の修正を促すことを行う．

4. 展開図の折りたたみ形状の推定

ORIPA の一機能として，前節で述べた「局所平坦条件」を満たす展開図について，折りたたみ後の形状を推定して表示する機能の実装を行った．本章ではそのために用いた手法を述べる．

4.1 多角形面素の取得

折り線または輪郭線で囲まれた閉領域を，以降では「多角形面素」と呼ぶこととする．また文脈から誤解のない場合には単に「面」と呼ぶこともある．

この多角形面素は，展開図中の折り線または輪郭線を**図 4** 中の F_0〜F_3 のように反時計回りに巡回して得られる．多角形面素には次のような性質がある．

- 多角形面素は互いに重なり合わない．
- 多角形面素の和は折紙の形に等しい．
- 折紙の輪郭が凸多角形であれば，すべての多角形面素は凸多角形である．

4.2 折りたたみ

本手法ではまず，多角形面素どうしの重なり順を考慮せずに，折りたたみ操作によって，それぞれの多角形面素が平面上でどの位置に移動するかを推定する．

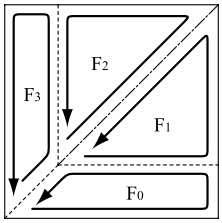

図 4 展開図からの多角形面素の取得
Fig. 4 Extraction of polygon-elements from a crease pattern.

このような推定は Shimanuki ら[10] によっても実装されており，特に新しい手法ではないが，本研究で用いた手法を以下に示す．

まずはじめに起点となる任意の多角形面素を決定し，それに隣接する多角形面素を再帰的に巡回する．隣接面に移動するごとに，2 面間を介する稜線によって（山折，谷折の別によらずに）多角形面素を折り返す（つまり平面上で折り線を介して反転させる）ことを行う．この反転操作は，各多角形面素に対してたかだか 1 回行うものとする．これにより，基点となる面から奇数回の移動で到達する場合は面の向きが裏に，偶数回の移動で到達する場合は面の向きが表になる．すべての多角形面素を巡回した時点で折りたたみ操作が完了する．

展開図が局所平坦条件を満たすのであれば，(基点の面には依存するが）巡回の順番には依存せずに一意の結果を得ることができる．

4.3 折りたたみの結果

上記の手法で，展開図から多角形面素を取得し，その折りたたみ後の位置を算出することで**図 5** の結果が得られる．(a) は鶴の展開図であり，(b) は折りたたみ後に面が上を向く多角形面素を色つきで表している．(c) は折りたたみ後の多角形面素の輪郭を表示したもの（輪郭線は太線で表示している），(d) は透過色で塗りつぶしたものである．これにより，面の向きが隣接する多角形面素で互いに異なること，および前述の方法で折りたたみ後の形状の算出が正しく行われていることを確認できる．なお，この時点では多角形面素の重なり順は未定である．

4.4 多角形面素の重なり順の決定

前節で述べた折りたたみ操作で，平面上での各多角形面素の位置を得ることができるが，それぞれの重なり順は求まらない．以降では，本稿で新しく提案する各多角形面素の重なり順を決定する方法を述べる．なお，**図 6** に示す「ねじり折り」のような，重なり順の関係が閉ループをなす作品は対象外とする．

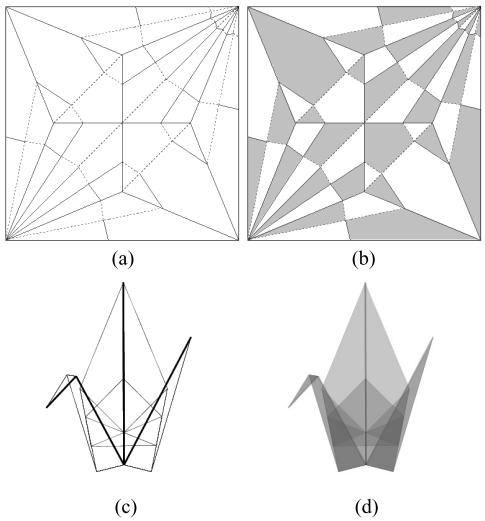

図 5 (a) ORIPA で作図した展開図，(b) 面の向きによって色分けされた展開図，(c) 多角形面素の輪郭，(d) 透過色で塗りつぶした多角形面素
Fig. 5 (a) Crease pattern designed with ORIPA. (b) Polygon-elements. Colored based on the direction. (c) Contours of polygon-elements. (d) Polygon-elements filled with transmitting color.

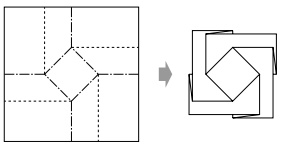

図 6 ねじり折り
Fig. 6 Twisting fold.

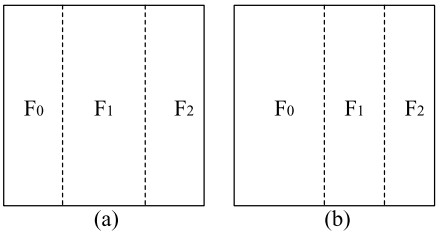

図 7 多角形面素の重なり順の制約（波線は谷折）
Fig. 7 Constraint for overlapping order (dot lines are for valley fold).

多角形面素の重なり順には制約があり，折り線に従って折り操作を行う場合，多角形面素どうしが衝突して実現できない重なり順が存在する．たとえば，**図 7** を例にあげると，F_1 を最下層に配置した場合 (a) の展開図では F_0 と F_2 のどちらが上でも問題ないが，(b) は F_2 が F_0 の上になることができない．この例から，多角形面素の重なり順を決定するには，展開図の位相情報からだけでは判断できず，多角形面素の形状に基

```
addFace() {
  foreach(FaceStack に含まれない Face f) {
  // f を FaceStack の末尾に追加してよいか判定
    if(FaceStack.canAddFace(f)) {
      FaceStack.push(f);
      if(全ての面の順番が決まったら) {
        処理を終了; // 妥当な解が見つかった
      }
      addFace(); // 再帰的に次の層へ
    }
  }

  if(FaceStack.empty()) {
    処理終了; // 妥当な解が見つからなかった
  } else {
    FaceStack.pop();
  }
}
```

図 8 多角形面素の重なり順の決定
Fig. 8 Algorithm for finding valid stacking orders.

づく幾何的な判定が必要があることが分かる．

面の重なり順を決定する問題は NP 困難な問題であることが文献 19) によって示されているため，本手法では，とりうるすべての重なり順を対象に探索を行い，折紙として妥当な重なり順であるものを抽出することとする．ところで，多角形面素の数を N とすると，その重ね順の場合の数は $N!$ 通りあり，N が増大するとすぐに場合の数は爆発してしまう．たとえば図 5 の鶴の例では多角形面要素の数が 52 であり，$52! \approx 10^{67}$ 通りの重なり順をすべて調べるのは現実的でない．

そこで，できるだけ判定の数を減らすために**図 8** に示すアルゴリズムを用いる．このアルゴリズムでは，初めに空のスタック FaceStack を用意し，そこに多角形面素を重なり順に格納することを行う．addFace 関数では自身を再帰的に呼び出すことで，順番に面をスタックに追加するが，面の重なり順として妥当でない面が追加された場合はスタックからポップし，次の面を格納することを行う．このように，早い段階で実現不可能な重なり順を枝刈りすることで，すべての重なり順を調べるより効率良く妥当でないものを判定の対象からはずすことができる．すべての多角形面素を問題なくスタックに格納し終わったら，妥当な重なり順が求まったものと判断できる．ただし，妥当な重なり順は複数ある場合があるので，解が求まった後もすべての判定が終わるまで処理は継続させる．

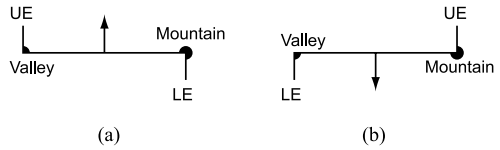

図 9 面の向きと接続（矢印は面の向きを表す）
Fig. 9 Direction of a face and face-connection (Arrows indicate face directions).

図 8 の addFace 関数では，スタックに面を追加してよいか否かを判定するための関数 canAddFace を呼び出しているが，この関数はスタックの最上位に，引数の多角形面素 F を追加できる場合に true を，そうでない場合に false を返すものとする．ここで，スタックに F を追加できる場合とは，F を追加しても次の 2 つの条件が満たされる場合である．

(1) 折り線での折り曲げ方が展開図の山折，谷折と矛盾しない．
(2) 任意の断面で面の交差が生じない．

上記の条件の判定方法を説明する前に，面の稜線は CE: Contour Edge（輪郭線），UE: Upper Connect Edge（上に接続する稜線）および LE: Lower Connect Edge（下に接続する稜線）に分類済みであるとする．面の表側が上を向いている場合，谷折の稜線は LE，山折の稜線は UE となる．面が下を向いている場合はこの逆となる（**図 9**）．面の表側が上下のどちらを向いているかは前章の折りたたみ操作で確定している．

折紙の多角形面素の重なり順に，スタックに下から順番に面を追加していくことを考えると，追加する面 F に LE を介して隣接する面 F' が，必ずスタック内に存在する必要がある（スタック内に F' が存在しない場合，F よりも F' が上方に配置されることになり，LE の谷折，山折が展開図と矛盾することとなる）．つまり，このことが上記 (1) の条件である．

(2) の条件の判定では，新しく配置する面 F がすでにスタックに格納されている面 F' の UE の接続を妨害しなければよい．各面 F' に含まれる未接続の UE（以降 e と表現する）を，面 F が覆い隠す位置にある場合（**図 10** (a)）は false となる（図 10 では，最上位の面を太線で表している．また灰色の領域は理解しやすくするために幅を持たせてあるが幾何的には幅ゼロの領域である）．e が面 F の UE と重なり合う位置にある場合（図 10 (b)）は true である．e が面 F の LE と重なり合う位置にある場合，面 F と隣接する面 $F_{neighbor}$ との位置関係によって条件が異なる．

- F' が $F_{neighbor}$ より下の場合（図 10 (c)）は true

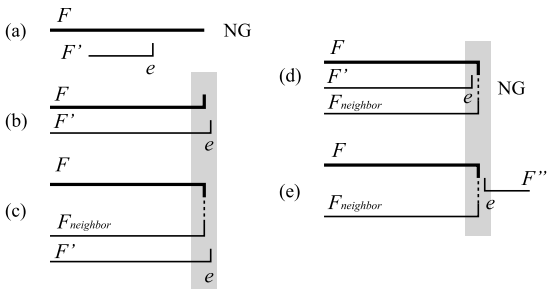

図 10 多角形面素の重なり順と稜線の関係
Fig. 10 Relations between polygon-elements and edges.

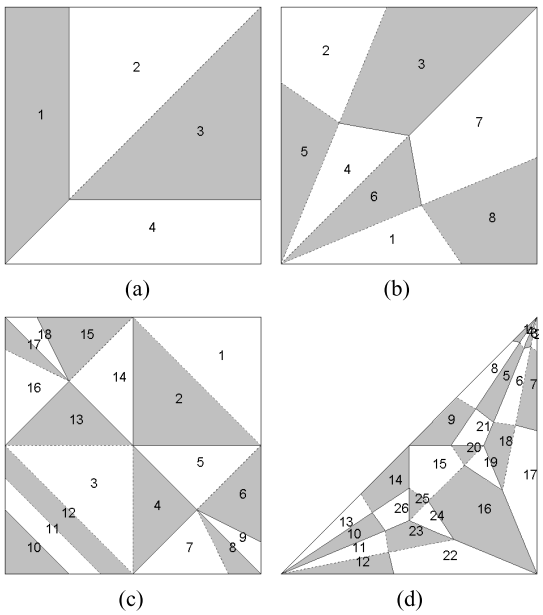

図 11 結果
Fig. 11 Results.

である．

- F' が $F_{neighbor}$ より上の場合で，F' と $F_{neighbor}$ が重なり合う場合（図 10 (d)）は false である．
- F' が $F_{neighbor}$ より上の場合で，F' と $F_{neighbor}$ が重なり合わない場合（図 10 (e)）は true である．

4.5 結　　果

3 章で述べた機能を実装したエディタを開発し，そのエディタで作成した展開図から 4 章で述べた手法で折りたたみ後の形状の推定および面の重なり順の推定を行った．用いた PC は CPU: Pentium Mobile Processor2.0 GHz，RAM1.0 GB である．対象とした展開図は**図 11** に示す 4 つで，それぞれ (a) は図 4 に示した単純な例，(b) は中割折りの例，(c) は兜，(d) は鶴の半分の展開図である．鶴に関しては展開図が対称な形状であることを考慮して，その半分の展開図に対して評価を行った．実験結果は**表 1** に示すとおりであった．表中の N は多角形面素の数，N' は面の重なりの妥当性の判定を行った回数，#Ans は得られた解の数，time は計算に要した時間（ミリ秒単位）である．図 11 中の多角形面素に振られている数字は，最初に見つかった解の面の重なり順を表している．

表 1 結果
Table 1 Results.

	N	N!	N'	#Ans	time(ms)
(a)	4	24	10	1	0
(b)	8	4.0×10^4	79	3	10
(c)	18	6.4×10^{15}	8.3×10^4	2778	2.2×10^3
(d)	26	4.0×10^{26}	3.4×10^6	5.0×10^5	1.2×10^5

5. 考　　察

5.1 エディタ機能

開発したエディタでは，折紙の展開図に特化した線分入力機能により，効率的に作図を行うことができた．操作に慣れると，図 11 (d) の鶴の展開図程度であれば 2〜3 分で作図することができた．

折紙の創作活動を行い，作品の展開図を公開している作家の多くは，折り図の作図に Illustrator や FreeHand などのドローソフトを用いることから，展開図の作図にもこれらのドローソフトを用いることが多い．CAD ソフトと異なり，これらのドローソフトには垂直二等分線や角の二等分線を効率的に入力するコマンドが存在しないことから，ORIPA を用いることで従来より大幅に作図時間を短縮することができる．実際に折紙の創作家に ORIPA を使用していただいたところ，展開図の作図がきわめて便利で作図時間が大幅に短縮できたこと，および展開図の妥当性チェック機能が折り線の入力漏れを防ぐうえで有効であることを報告いただき，良い評価を得ることができた．ORIPA で作図した展開図は，DXF 形式でドローソフトに読み込むことが可能なため，使い慣れた従来のソフトウェアで追加の編集も可能である．

5.2 折りたたみ後の形状の推定機能

折りたたみ後の形状の推定については，例題にあげた程度の複雑さを持つ展開図であれば，実用的な時間で面の重なり順の解をすべて数え上げることができた．中割折りを行った単純な展開図である図 11 (b) については，全部で 3 通りの異なる折り方があることが判明し，興味深い結果を得ることができた（図 11 の解は，中割折りではない折り方の例である）．

兜の例では 4 万近くの解が抽出されたが，これは互いに重なり合わない面の対が多数存在する場合，外

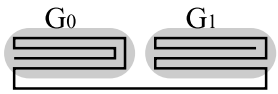

図 12 互いに関与しない面の重なり
Fig. 12 Polygon-elements with no effect to result.

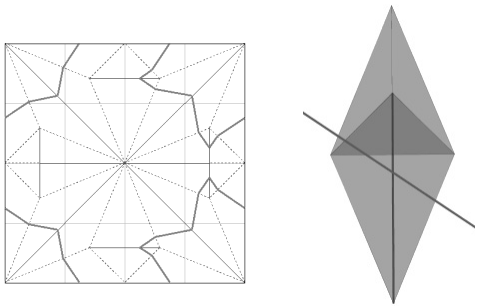

図 13 折紙を切断した結果のシミュレート
Fig. 13 Simulation of cutting a folded Origami.

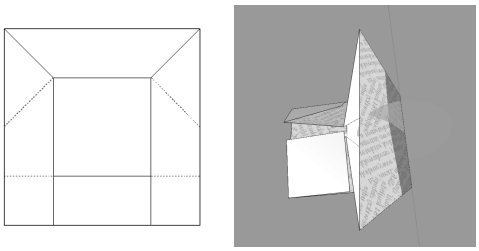

図 14 PC 上での対話的な折りたたみ操作[20)]
Fig. 14 Interactive Origami folding on a PC [20)].

見が同一となる解が増大するためである．たとえば，**図 12** の例では，左右の多角形面素をそれぞれグループ G_0，G_1 に分け，それぞれの面の数を N_0，N_1 とした場合，${}_{(N_0+N_1)}C_{N_0}$ 通りの，外見的に同一のものが生成されることになる．これは，異なるグループに属する面との重なり順の前後は得られる形状に影響を与えないためである．兜の例で，実質的に異なる解は著者自身が実際に紙で確認したところ 9 通りであった．

鶴の例では，多角形面素の数が 26 であり，すべての場合は実時間で計算しきれないが，効率的に不要な判定を除くことができた結果，判定処理を行った数は 3.4×10^6 に収まっている．しかし，この例でも兜の例と同じ理由で解の数が膨大になっている．今回の実装では，折りたたみ後の面の重なり順として妥当なものを網羅することはできるが，外見が異なるケースが何通りあるかを数え上げるには，そのための手法を新たに考案する必要がある．

5.3 他の応用アプリケーションへの活用

ORIPA は Web で一般公開しており，これを活用した研究がすでにいくつか行われているため，ここ

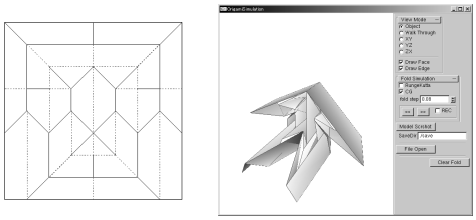

図 15 剛体折紙の折りたたみのシミュレート[21)]
Fig. 15 Simulation of Rigid Origami [21)].

ではそれらの事例をあげる．**図 13** は折紙を直線で切断したときに，その切断線が展開図上でどのように配置されるかをシミュレートしたものであり，**図 14** はユーザが対話的に折り操作を行うインタフェースの研究[20)]，**図 15** は折紙剛体折りのシミュレーションの研究[21)] の様子である．

このように折紙の展開図のエディタとしての機能を提供し，ファイルフォーマットをオープンとすることで，今後の折紙の研究へ活用することが期待できる．

6. まとめと展望

本稿では折紙の展開図の入力に特化したエディタ，ORIPA の開発を行い，それを用いて作図した展開図から，折りたたみ後の形状を推定する手法を提案した．また，ORIPA を用いた複数の研究事例をあげ，研究の基盤ツールとして ORIPA が活用できることを示した．ただし現在の ORIPA はすでに形が定まっている折紙作品の展開図の入力には便利であるが，新しい作品を試行錯誤しながらデザインすることを支援する目的には向いていない．展開図の作図と連動してリアルタイムで折りたたみ後の形状を出力するなど，新規作品の創出を支援するツールとして発展させることが今後の課題である．また，本稿で提案した折りたたみ後の形状を推定する手法については，展開図から折りたたみ後の形状を構築できることを示せたが，さらに探索の数を減らし，いかに高速化するかが課題である．たとえば，図 12 に示した例のように，互いに影響を与えないグループを効率的にまとめることで高速化が図れると考えられる．

参 考 文 献

1) 小松英夫：馬，折紙探偵団，Vol.10, No.6, pp.22–32 (2000).
2) 川崎敏和：バラと折り紙と数学と，森北出版 (1998).
3) 深川英俊：折紙の数学，森北出版 (2002).
4) Lang, R.J.: *Origami Design Secrets: Mathematical Methods for an Ancient Art*, AK Pe-

ters, Ltd. (2003).
5) 内田 忠，伊藤英則：折り紙過程の知識表現とその処理プログラムの作成，情報処理学会論文誌，Vol.32, No.12, pp.1566–1573 (1991).
6) Miyazaki, S., Yasuda, T., Yokoi, S. and Toriwaki, J.: An Origami Playing Simulator in the Virtual Space, *The Journal of Visualization and Computer Animation*, Vol.7, No.1, pp.25–42 (1996).
7) 宮崎慎也：折り紙シミュレーション. http://www.om.sccs.chukyo-u.ac.jp/main/research/origami/indexj.html
8) Kato, J., Watanabe, T., Hase, H. and Nakayama, T.: Understanding Illustrations of Origami Drill Books, 情報処理学会論文誌，Vol.41, No.6, pp.1857–1873 (2000).
9) 島貫 博，加藤ジェーン，渡邉豊英：展開図を用いた折り紙操作過程における手順毎の折り方の構成，電子情報通信学会技術研究報告，Vol.102, No.55, pp.71–78 (2002).
10) Shimanuki, H., Kato, J. and Watanabe, T.: Construction of 3-D Paper-made Objects from Crease Patterns, *Proc. IAPR Conference on Machine Vision Applications* (*MVA2005*), pp.35–38 (2005).
11) 河合良太，西川 玲，佐賀聡人：手書きスケッチ入力フロントエンドプロセッサ：SKIT，電子情報通信学会論文誌，Vol.J88-DII, No.5, pp.897–905 (2005).
12) Huzita, H.: Understanding Geometry through Origami Axioms, *Proc. 1st International Conference on Origami in Education and Therapy* (*COET91*), pp.37–70 (1992).
13) Hatori, K.: Origami Construction. http://origami.ousaan.com/library/conste.html
14) Lang, R.J.: Origami and Geometric Constructions. http://www.langorigami.com/science/hha/origami_constructions.pdf
15) 三谷 純：折紙展開図エディタ ORIPA. http://mitani.cs.tsukuba.ac.jp/pukiwiki-oripa/
16) Kawasaki, T.: On the Relation Between Mountain-creases and Valley-creases of a Flat Origami, *Proc. 1st International Meeting of Origami Science and Technology*, pp.229–237 (1989).
17) Justin, J.: Towards a Mathematical Theory of Origami, *Proc. 2nd International Meeting of Origami Science and Scientific Origami*, pp.15–29 (1997).
18) Hull, T.: On the Mathematics of Flat Origamis, *Congressus Numerantium*, Vol.100, pp.215–224 (1994).
19) Bern, M. and Hayes, B.: The complexity of flat origami, *Proc. 7th annual ACM-SIAM symposium on Discrete algorithms*, pp.175–183 (1996).
20) 古田陽介，三谷 純，福井幸男：折紙の展開図情報を入力とした対話的な折り操作のシミュレーション，日本図学会 2006 年度本部例会学術講演論文集，pp.33–38 (2006).
21) Tachi, T.: Simulation of Rigid Origami, *to appear in proceedings of the 4OSME* (2006).

(平成 18 年 9 月 27 日受付)
(平成 19 年 6 月 5 日採録)

三谷 純（正会員）

1975 年生．2004 年東京大学大学院工学系研究科博士課程修了，独立行政法人理化学研究所基礎科学特別研究員，2005 年筑波大学大学院システム情報工学研究科講師，現在に至る．コンピューターグラフィックスおよび計算機を用いたペーパークラフトの設計支援に関する研究に従事．博士（工学），科学技術振興機構さきがけ研究員，日本図学会，芸術科学会，画像電子学会各会員．