# 修 士 論 文 の 和 文 要 旨

| | | | |
|---|---|---|---|
| 研究科・専攻 | 大学院 情報システム学研究科 情報メディアシステム学専攻 博士前期課程 | | |
| 氏　　名 | 藤岡 直幹 | 学籍番号 | 1450030 |
| 論 文 題 目 | 折り紙公理に基づく谷折り操作の記述・認識に関する研究 | | |


要　旨

日常生活には，多くの柔軟物体が存在するが，把持や操作を行うごとにその挙動が異なることから，作業をロボットに指示することは困難である．ロボットにより人間のように巧みな柔軟物体の操作が実現すれば，工場等から介護や福祉，家庭環境などに普及していく一助となることが期待される．本研究では，紙を谷折りする作業を対象として，人間によって行われた谷折り操作を認識し，ロボットの作業記述に適した形で記述する手法を提案する．

そのために，折り紙作業を逐次的な「操作の連続と見なし」，操作前後の紙の状態と実行された操作の内容を明確にする記述方法を提案し，さらに人間が行う折り操作からその記述に必要なパラメータを推定する手法を提案する．

まず，折り紙公理の折り紙の線や点の扱いについて分析し，どのような情報が折り紙公理に基づく作業記述にとって必要か示した．それに基づき折り紙公理で作業を記述するためのデータ構造を定義し，それを用いて折り紙作業を記述する手法を示した．

次に，折り線検出に基づく折り操作の認識手法の提案および評価を行った．ここでは，折り操作前後の画像から折り線を画像認識により推定した．その後，行われた操作の種類と折り動作後の折り紙の状態を推定する手法を提案した． 提案した手法に対し実験により評価を行い，この手法により二回折りまでの折り作業が推定可能であることを示した．

続いて 上述の手法で対応できない折り方に対応する手法を検討した．この手法では，まず折り動作前の折り紙の状態に対し，折り紙公理から記述可能な全ての折り線を列挙する手法を提案した．その際，折り線には適用された公理の情報を保存することで折り線自体に作業としての情報を持たせた．その後，実際の折り動作後の形状に最も近い折り線を選び出し，折り動作後の折り紙の状態を推定する手法を提案した．示した手法に対して実験を行い，三回折りまでの折り作品に対して，認識可能であることを示した．また，認識失敗した例から，このアルゴリズムが点や線の増加により認識率が低下するが示唆されたため，それに対応するための今後のアルゴリズムの改良についての検討を行った．

平成 27 年度修士論文

# 折り紙公理に基づく谷折り操作の記述・認識に関する研究

大学院情報システム学研究科
情報メディアシステム学専攻

学 籍 番 号　：　1450030
氏　　名　：　藤岡 直幹
主任指導教員　：　工藤 俊亮 准教授
指 導 教 員　：　末廣 尚士 教授
指 導 教 員　：　田野 俊一 教授
提出年月日　：　平成 28 年 01 月 28 日（木）

# 目 次

# 図 目 次

# 表 目 次

# 第1章　初めに

## 1.1　研究背景

近年のロボット技術の進歩により，ロボットは従来と異なり，工場での労働力としてだけでなく，日常生活の中での生活支援に対する期待が高まっている．そういった状況に反し，日常生活の中でロボットが活動することを考えた場合，未だ多くの問題が存在する．問題の1つとして，日常生活の中に多く存在する紙や布といった柔軟物体の取り扱いがある．柔軟物体は，把持や操作を行うごとにその挙動が異なることから，作業をロボットに指示することが困難である．そのほかにも，柔軟物体は形状が複雑に変形しうることからその形状をとらえることも困難となる．これらの理由から，未だ工場におけるロボットでも柔軟物体を扱うロボットの多くは特定の作業のみに特化した物であるのが現状である．

もし，ロボットにより人間のように巧みな柔軟物体の操作が実現すれば，介護や福祉，家庭環境などに普及していくだけでなく，工場においてもロボットが活躍するフィールド拡大の一助となることが期待される．

そういった今後期待される日常生活での柔軟物体操作の１つに，紙の操作が存在する．紙は我々の生活の中でも欠かせないものであり，特に日本では古来より折り紙が親しまれている．現代では日本だけでなく世界中で「Origami」などと呼ばれ，様々な世代に親しまれている．親しまれる理由の１つとして，特に谷折りなどに絞った場合，折り紙が点や線などを単純に重ねていくことである程度作品を作れることが考えられる．ロボットにもそういった単純な指標を与えることで折り紙の操作を指示できる可能性が考えられる．

本研究ではこれら折り紙の特性を用いることでロボットの作業記述が可能であると考え，折り紙作業の研究を行うこととした．

## 1.2 関連研究

近年，ロボットにより柔軟物体を操作する研究は盛んであり，高い成果が挙げられている．本研究ではその中でも主に 2 つに注目する．それらはロボットによる操作のためのものと，柔軟物体の形状を推定する物である．以下にはそれら 2 つの方向それぞれに対し関連研究を述べる．

1. ロボットによる操作

   紐に関する研究では Vinh らは，紐結びの動作教示を行い，その教示の中からキーポイントの抽出を行うことで止め結びの実現を行っている [1] [2] [3]．また，Takizawa らは紐をモデル化し，手先軌道を生成することでロボットによる卓上の任意曲線状に配置することの実現を行っている [4] [5]．布に関する研究では Hayashi らは目標線を用いた操作記述を行うことでロボットによる布被覆操作の研究を行っている [6]．また，Yamakawa らは高速ハンドシステムと高速視覚フィードバックシステムを用いることで布の動的折畳動作実現を高速に実現している [7] [8] 紙に関する研究では，Ueda らは紙をめくる操作を強化学習を用いて実現している [9]．特に折り紙操作に関する研究では，Yokokohji らによる研究が多くなされている．そこでは折り紙作業のためのロボットハンドの設計や，折り紙ロボットの人間による直接教示動作に基づく動作生成などを行うことで実際に「おたまじゃくし」などの作品をロボットにより完成させている．[10] [11] [12] [13]．これらの実際にロボットによる操作に関する研究に対し，別のアプローチとして数学的な研究もなされている．純粋な数学の研究として Fujita らや Hatori らにより提唱される 7 つの折り紙公理が存在している [14] [15] [16]．また，折り紙公理は，Lang により 7 つの公理全てがあれば，点と有限の線分を組み合わせた並びから構成される単純折りならどんなものでも定義できるという意味で完全であることが証明されている [17]．折り紙公理では，点や線を単純に折ることで構成されるものを正確にとらえることを意図している．例えば，折り紙公理の 1 つでは 2 つの点を重ね合わせることで折り線を生成するなど，人間が折る際に指標にしているものとして記述されている．しかし，この研究は純粋に線と点により折り線と折り返された状

態を幾何学的に計算することに特化しており，実際の操作に結びつくような記述はされていない．そのほかのものとして，Tsuruta らは，折り紙作品を作成する際の手順を示す折り図作成を補助するために，折り紙公理に基づいた手順予測を行う手法を提案している [18]．また，Mitani らによる折り紙の展開図を折り紙を折った際の展開図作成に特化したエディタである「ORIPA」を開発している [19]．本研究の先行研究として Maeda らにより折り紙公理に基づいたロボットによる折り紙作業記述を行った研究がある [20] [21]．ここではロボットに作業可能な形で折り作業を記述しているが，折り紙の状態を観察して作業に必要なパラメータを取得しているわけではない．

2. 柔軟物体の形状推定

   Schulman らはポイントクラウドのデータを用いて柔軟物をトラッキングする手法を実現している．[22] 紐の状態を推定する手法として橋本らは 2 リンクモデルとして 2 次元平面でモデル化を行い，紐のパラメータ道程による紐操作を提案している [23]．須藤らは，現実物体の観測画像することで柔軟物体のモデルパラメータを自動取得し，面上柔軟物体の静止状態を再現する研究を行っている [24]．前述の Mitani らは 2 次元バーコードを用いて折り紙の折りたたみ構造の認識とモデル化についても実現している [25]．折り紙シミュレータに関する研究では Miyazaki らや Yoshida らにより開発され，簡単な作品を計算機上で実現している．[26]，[27] しかし，これらは折り紙の状態を表しているものではあるが，ロボットの作業記述に適した形で記述されているわけではない．

## 1.3 研究目的

本研究では，紙を谷折りする作業を対象として，人間によって行われた谷折り操作を認識し，ロボットの作業記述に適した形で記述する手法を提案する．そのために，折り紙作業を逐次的な「操作の連続と見なし」，操作前後の紙の状態と実行された操作の内容を明確にす

る記述方法を提案し，さらに人間が行う折り操作からその記述に必要なパラメータを推定する手法を提案する．1.2 であげた数学的な関連研究では，人間により計算機上で教示される多種多様な折り紙作品に対し，計算機上で状態を表現し，作業を折り図という形で記述しているが，実際の折り動作に結びつく形で記述しているわけではない．先行研究として挙げた [20] [21] では，折り紙公理に即した形で折り作業に基づく作業記述を行っているが，折り紙の状態の観察などを行っていないため，作業に必要なパラメータなどは人の手により与えられていた．つまり，関連研究では「折り紙の状態や作業の観察及び表現」と「ロボットの作業記述に適した形での記述」が融合しておらず，また実際の折り紙に対する観察も行われていないというのが現状である．そこで，本研究では実際の折り紙に対し逐次的な作業内容と折り紙の状態の記述方法を明確にし，記述した作業や状態に必要なパラメータを推定する手法を提案する．

まず，折り紙公理に基づいた折り紙の作業記述及び状態の記述手法を提案する．また，折り紙公理では作業に直接結びつく形では記述されていないため，[20]，[21] の論文で記述されている折り作業の記述を拡張した記述方法についても述べる．次に，提案した記述手法で折り紙の作業と状態に必要となるパラメータを求めるアルゴリズムについて記述する．アルゴリズムは今回，2 つの側面から検討することした．1 つは，折り線認識に基づく操作の認識手法，もう 1 つを折り線候補群を用いた認識手法として検討する．その中で，1 回の折り動作ごとに折り紙の状態がどのように変化したかを認識する手法と，どの折り紙公理が適用可能であるかを認識する手法を提案する．

## 1.4 論文構成

本論文の構成は以下のとおりである．まず，第 2 章にて谷折操作の記述について記述する．第 3 章では，折れ線検出に基づく折り操作の認識手法について記述する．第 4 章では，折り線候補群を用いた認識手法について記述する．第 5 章では，本稿のまとめを記述する．

# 第2章　折り紙公理に基づく折り作業の記述

## 2.1　折り紙公理

折り紙を人間が行う際に，点と点を合わせるなどといった単純な操作を積み重ねていく場合が多くある．そういった折り紙の作業により作られる折り線を数学的に記述する手法として，7つの折り紙公理というものがある [14]．折り紙公理は，平面で完結，すべての折り線は直線という仮定の下に，既に存在する点や線から構成しうる新しい折り線をとらえることを可能としている．その例として図 2.1 に折り紙公理を図示し，以下に概要を記述する．

- 公理1：与えられた2点を通る線で折る．
- 公理2：与えられた2点を重ねて，2点間の垂直ニ等分線を折り線として折る．
- 公理3 ：与えられた2本の線を重ねて2線間のニ等分線を折り線として折る．
- 公理4 ：与えられた1点と1本の線に対し，点を通るように線を自分自身の上に重ねて，与えられた点を通るように与えられた線の垂直ニ等分線を折り線として折る．
- 公理5 ：与えられた2点と1本の線に対し，一方の点が一方の線に乗るように他方の点を通る線を折り線として折る．
- 公理6 ：与えられた2点と2本の線に対し，一方の点が線に乗るように他方の点を通る線で折る．
- 公理7 ：与えられた1点と2本の線に対し，その点が一方の線の上に乗るように他方の線の垂線を折る．

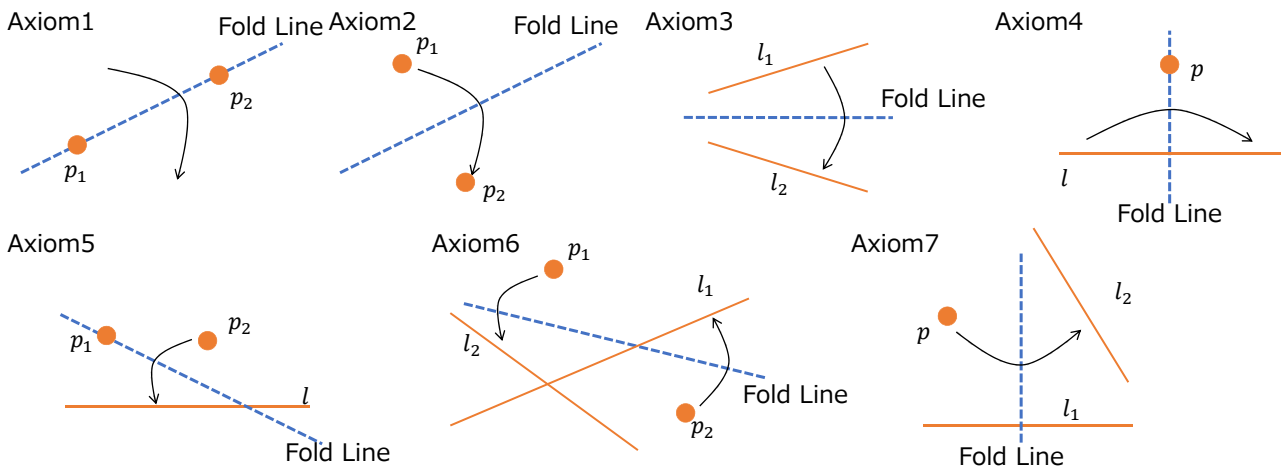

Axiom1
Fold Line
$p_2$
$p_1$
Axiom2
$p_1$
Fold Line
$p_2$
Axiom3
$l_1$
Fold Line
$l_2$
Axiom4
$p$
$l$
Fold Line
Axiom5
$p_1$
$p_2$
$l$
Fold Line
Axiom6
$p_1$
$l_1$
$l_2$
Fold Line
$p_2$
Axiom7
$l_2$
$p$
$l_1$
Fold Line

図 2.1: 折り紙公理

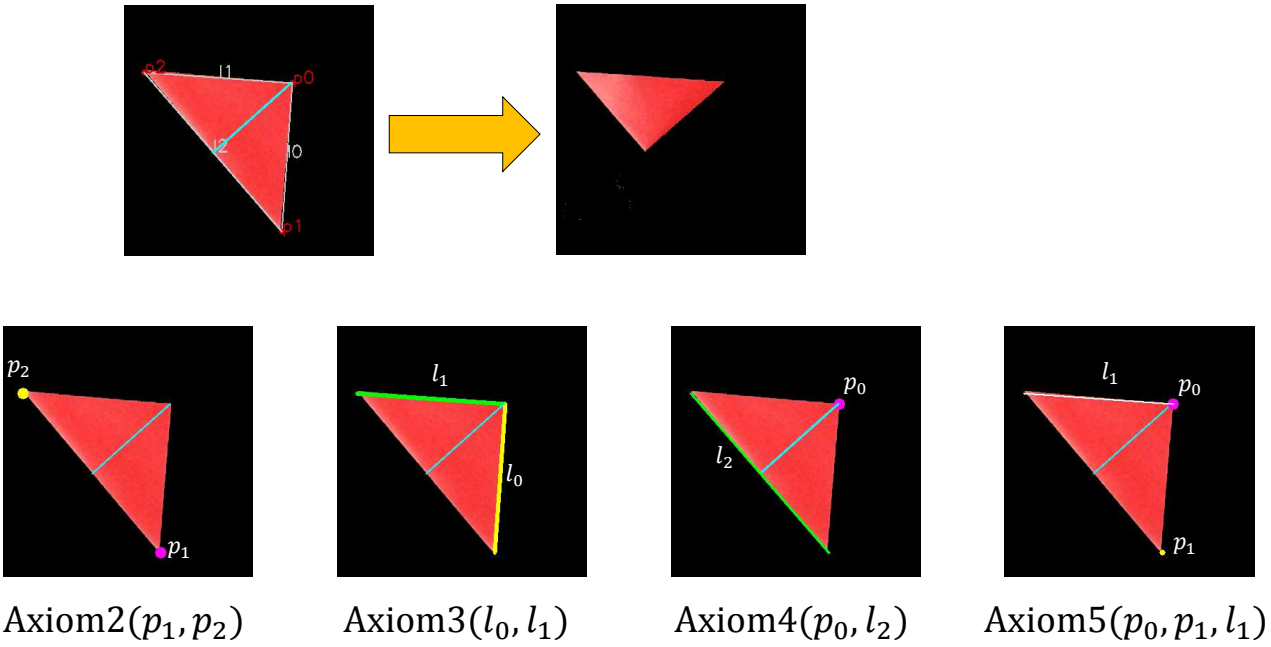



図 2.2: 複数の公理が適用可能な折り

ここで扱われる点や線，折り線とは以下のように記述される．

- 線：

  （有限サイズの）1 枚の紙の上の折り線や境界

- 点：

  2 つの線の交点

点や線はオクルードされ，ある一面から見えない場合にも点や線として使用可能である．また，1 つの折り線を生成可能な折り紙公理は複数存在可能などといった側面も持つ．例として，図 2.2 のような折り方では，1 つの折りに対して複数の公理が適用可能である．

これらのように折り紙公理では紙の点や線などの特徴を重ね合わせるという，作業逐次的な仕方で操作を記述している．この点において，折り紙公理は折り操作の記述の基礎として

の適性があると考える．また，ロボットに折り作業として与える際にも，このような基本的な記述であればロボットによる折り動作の教示にもつなげやすいと考えられる．よって，本研究では人間によって行われた折り作業をロボットにより観察し，折り紙公理に即した形で記述，認識する手法を検討する．

## 2.2 データ構造としての折り紙作業の記述

本節では上記折り紙公理を考慮した，折り紙作業を記述する手法について記述する．以下ではまず 1 回当たりの作業の記述方法について述べ，その後それらを統合して 1 つの一連の作業として記述するための手法を記述する．

### 2.2.1 折り紙に対して行われた作業の記述手法

本節では，2.1 で記述された折り紙公理に即した形での折り作業の記述手法について検討する．まず，2.1 の折り紙公理では作業につながる形では記述されていないため，[20] を拡張した以下のような形で折り紙の操作を関数として記述する．

折り紙公理に基づいた折り線の記述 Axiom_Info

Axiom1($p_1, p_2$)

2 点 $p_1$, $p_2$ から折り線を定義.

Axiom2($p_1, p_2$)

2 点 $p_1$, $p_2$ から折り線を定義.

Axiom3($l_1, l_2$)

2 線 $l_1$, $l_2$ から折り線を定義.

Axiom4($p_1, l_1$)

1 点 $p_1$，1 線 $l_1$, から折り線を定義.

Axiom5($p_1, p_2, l_1$)

2 点 $p_1$, $p_2$，1 線 $l_1$ から点 $p_1$ を通る折り線を定義.

Axiom6($p_1, p_2, l_1, l_2$)

2 点 $p_1$, $p_2$，2 線 $l_1$，$l_2$ から折り線を定義.（この際 $p_1$ を $l_1$ に重ね，$p_2$ を $l_2$ に重ねるものとする.）

Axiom7($p_1, l_1, l_2$)

1 点 $p_1$，2 線 $l_1$, $l_2$ から折り線を定義.（$p_1$ が $l_1$ に重なるものとする）

これら折り紙公理の記述を関数として記述した場合，どの公理を用いるのかが既知であれば，各公理ごとに必要となる点や線の情報は引数の順番のみで判断することが可能となる.（例として公理 4 では第 1 引数が点であり，第 2 引数が線であるという情報を保持することができる.）この際，特に，各公理においていくつかの条件を検討した.

公理 1，4

公理 1，4 では公理の記述上，折る方向の指定はできないが，できる作品は裏返すだけで同じものとなるため，今回これは考慮しないものとした.

公理 2，3

公理 2，3 では，引数の順番は考慮しないものとする.

公理 5

公理 5 では，線を点に重ねるか，点を線に重ねるかといったことが考慮できるが，今回はこれは考慮しないこととした.

その特性を考慮し，以下のような構造で公理 1 つ当たりの折り作業を記述することを検討した.

折り紙公理構造体 Axiom_Info

適用された折り紙公理の番号： 整数　orisen_vertex_ids

適用された折り紙公理に使用された辺や頂点の ID： 整数の動的配列 axiom_infos

また，折り紙公理は1つの折り作業に対して複数適用することが可能な場合が多く存在する．そのため，1つの折り作業をデータとして記述する際には複数の折り紙公理による作業を記述可能にしておく必要がある．このようにすることで，ロボットが実際におる際にロボットにとって成功率の高い公理による折り方を選ぶことが可能になることも期待する．また，折り紙公理による定義であるが，公理3などにおいて，与えた点や線の情報のみでは折り線が2つ発生する場合が存在する（詳しくは後述参照）．そのため，検出した折り線を保存しておく必要も存在する．以下にそれらを考慮した1つの折り作業当たりの構造体を記述する．

折り操作構造体 Folding_Operation

折り線の通る頂点ID： 整数の動的配列 orisen_vertex_ids

適用される折り紙公理の情報： Axiom_Info 構造体の動的配列 axiom_infos

上記のように記述することで，ロボットに作業可能な形での記述とする．

### 2.2.2 折り紙の状態記述手法

折り紙をデータとして記述し，認識することを検討した場合に問題となるのは，折り紙が折り曲げられる位置などは一定ではないため，行われた作業に応じて柔軟な記述をしなければならないという点である．また，折り紙公理で作業を記述することを検討した場合いくつかの制約が考えられる．本研究では，これらを検討し，対応を可能とした折り紙の状態モデルを提案し，そのモデルをもって折り紙の状態記述を行う．
検討した折り紙モデルの要件として以下の事柄を検討する必要があると考えた．

- 折り紙公理での点や線の記述に即したモデルの記述
  折り紙公理で必要となる情報は点と線の情報のみであり，折り紙公理では平面であることを仮定しているため座標情報は全て2次元の情報として与えられる．しかし，実際の折り紙の作業は重ね合わせて行われていくものであり，必然的に折り紙は3次元的な構造を持つことになる．そのため，折り紙公理で検討する際には3次元の複数平面

となる折り紙を，2次元の一平面として折り紙を扱う必要性が生じる．また，折り線の記述も2次元で記述される問題も存在する．これらを検討した結果，本研究で検討するモデルは，3次元的に存在する折り紙の構造を，一平面として記述することとした．点や線の記述としては，点は線の交点として記述されることから，これらはグラフにおけるノードとエッジとしてみることが可能である．そのため，本研究では線は2つの点により構成されるものとした．そのために線には2つの点の座標情報を持たせずに点のIDを保持するものとした．また点は座標情報とIDの情報を持つものとした．

- オクルージョンの発生に対応可能なデータの記述方法

  オクルージョンに対応するため，折り紙の構造を初期値を既知として与え（不切正方形一枚折りの場合正方形の状態），そこからの変化を累積していくことでオクルードされた領域にあるものも推定する手法を今回検討するものとした．1回の折り動作ごとに，折り紙の新たな状態構造体を1つの平面として生成することを目指す．また，折り動作ごとに得られた状態を連続した状態として保存し，過去の状態を参照可能な構造を検討する．そうすることで，ある点が最初の状態で何の点であったのかといった情報を追うことが可能となる．そういったことから，本研究では，生成した点や線に対し，1つ前の状態ではどのIDであったかという情報を保持するようにすることとした．

- データを記述する座標系

  折り紙を点と線で記述すると仮定した場合，次の3つの座標系で検討する方法が考えられた．以下にそれらを記す．

  1 撮影された画像座標系の任意の点を折り紙座標系の原点とする．

  2 折り作業ごとに折り紙の適当な1点を原点とし，その1点を折り動作ごとにトラッキングする．

  3 折り紙の初期状態の正方形の状態時に1つの頂点を原点とし，折り動作が行われても移動しないものとする．

1の方法では，折り紙の記述は折り紙の位置に依存するため，折り紙の状態のみを記述

しているとは言えない．また，2 では，オクルージョンが発生し，トラッキングができなくなった際などに破綻することが考えられる．また，折り紙が折り返された際などに折り紙座標系が反転するなども考えられる．3 では，常に折り紙の頂点またはその付近 1 か所を座標原点とすることができ，折り紙が裏返されてた際にも，折り紙を折り紙座標中で裏返すというだけで対応できる．また，ロボットによる作業を記述する際に，折り紙が裏返りや回転するなどするたびに作業座標系が反転，回転することを考慮してしまうと，記述が複雑になるため望ましくない．そのため本研究では 3 の方法による折り紙座標系の記述方法とした．

以下に上記を考慮した点と頂点のデータ構造を記述する．

頂点情報構造体 Vertex

座標情報： 座標情報 pos

1 つ前の状態での ID： 整数の動的配列 old_ids;

辺情報構造体 Edge

辺の頂点の ID： 整数の 2 次元配列 vertex_ids[2]

1 つ前の状態での ID： 整数の動的配列 old_ids

これら点と線の構造体を折り紙の状態として構成するための構造体を以下に示す．

折り紙状態構造体 State

全ての頂点の情報を格納する構造体： 整数のキー値と Vertex 構造体の値を持つ辞書型配列 vertices

全ての辺の情報を格納する構造体： 整数のキー値と Edge 構造体の値を持つ辞書型配列 edges

折り紙の初期状態の 1 辺の長さ： 整数　origami_first_edge_size

上記の origami_first_edge_size とは，折り紙の展開した状態の 1 辺の長さを保持するものとして記述される．このようにして記述することで，様々な大きさの折り紙に対応できるほか，折り紙に限らず比が求まっているものであればどのような大きさの紙の記述にも対応可能なことを想定している．

### 2.2.3　連続した折り紙作業の記述

折り紙の状態の初期値を既知とし，作業が行われるたびに作業後の新たな折り紙の状態と，折り紙に対して行われた操作の認識を行い，折り紙の状態を更新する．また，その際に折り紙の状態と作業内容は作業が行われる順に保存されていき．そのため，折り紙の状態及び作業を連続した状態で保持する構造が必要となる．そこで，これら折り紙の状態や作業の遷移を折り作品を作るための一連の流れ（タスク）として記述することとした．そのために検討したのが以下の構造体である．

タスク構造体 Task

状態構造体の配列： State の動的配列 States

操作構造体の配列： Folding_Operation の動的配列 Folding_Operations

以上のように記述することで，1 つの折り作品を作る作業を 1 つのタスクとして保持することとした．また，このように記述することで，作業を途中から行うことが可能になるほか，途中から別の作業を記述開始できることも期待する．

## 2.3　折り紙で作成可能な折りのラベル付け

折り紙に対し，折り紙公理で 2 回折りまでに作成可能な折り紙を全てと，3 回折りで生成可能な一部に対して調査し，ラベル付けを行った．2 回折りまでをラベル付けした結果は図 2.3 以下である．3 回折りまでをラベル付けした結果は図 2.4，図 2.5 に示す．今回検討した

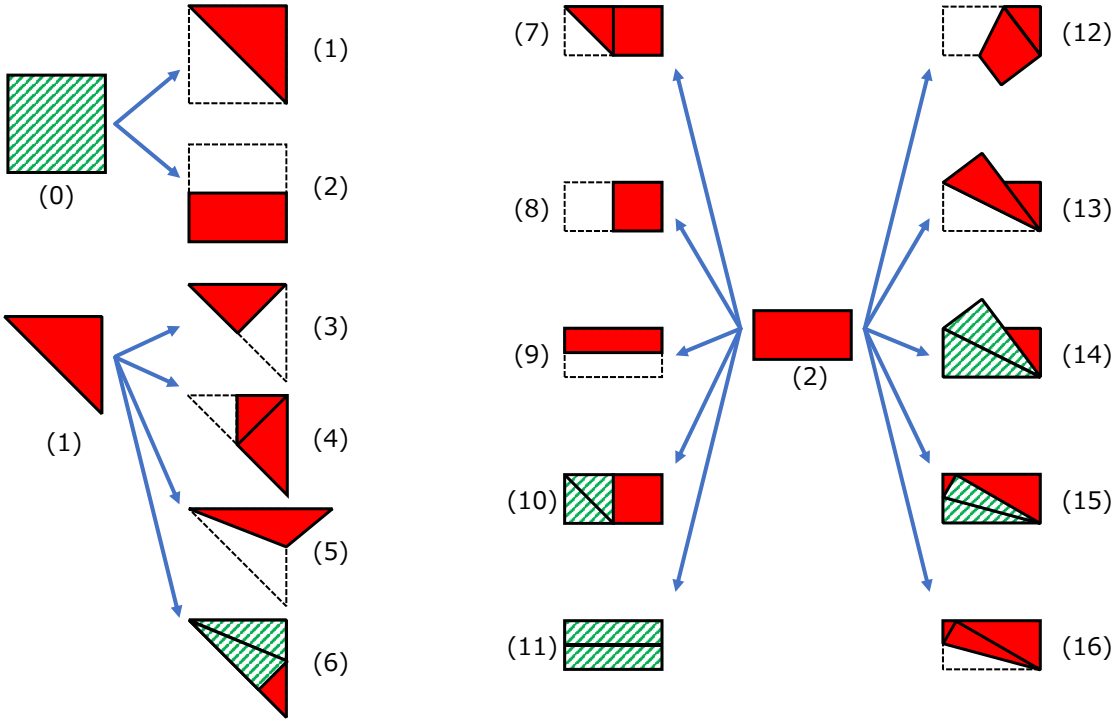



図 2.3: 2 回折りまでの折り作品

折り紙は，回転，反転，裏返しを行うと同じになるものは排除するものとした．3 回折りでは (13)，(15)，(4) および (6) にたいして可能なそれぞれ 5 種類を検討した．

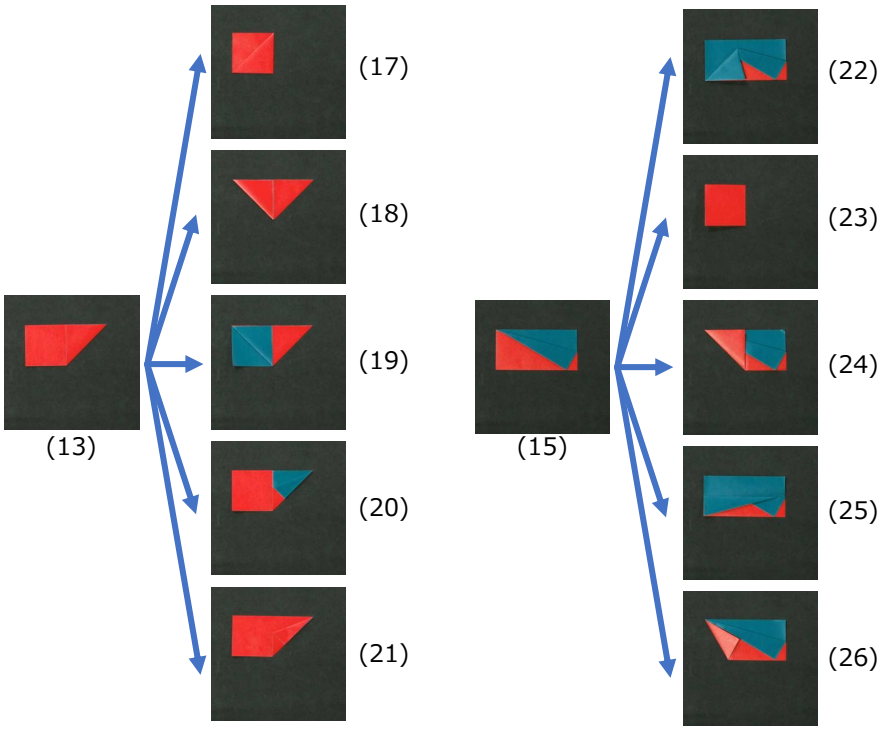



図 2.4: 3 回折りで可能な折り作品 1

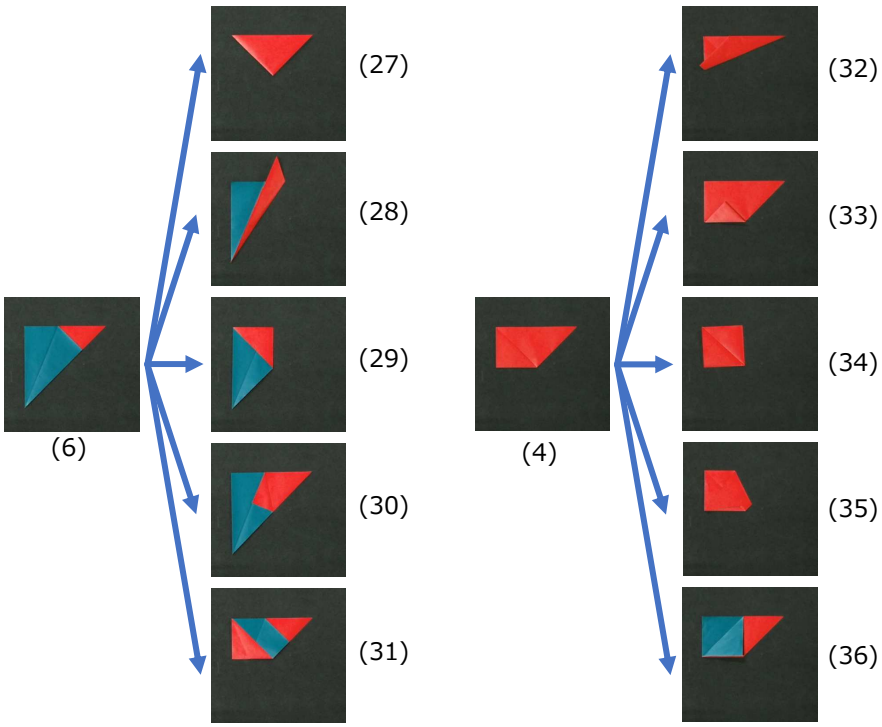



図 2.5: 3 回折りで可能な折り作品 2

# 第3章　折れ線検出に基づく折り操作の認識手法

本章では，画像処理により折り線の位置を検出し，折り線から折り操作の認識を行う手法を提案する．以下では，まず折り紙の画像を撮影するシステムについて記述する．その後，アルゴリズムの概要と詳細を記述する．最後に検討したアルゴリズムに対し実験を行う．

## 3.1　折り紙画像の撮影システム

本アルゴリズムでは折り紙の色画像の情報を観測する必要がある．本研究で使用する画像取得のための撮影システムを図 3.1 に示す．

カメラにはKinectv2を使用し，折り紙を置く撮影台には4か所の黄色のマーカを配置している．本研究で用いる色画像はKinectv2により撮影された画像を，マーカを用いて切り抜いて作成する．また，マーカの検出は色抽出により行われる．図 3.2 にカメラにより撮影された画像を，図 3.3 に切り抜かれた画像を例として示す．以降ではこのうち，切り抜かれた画像を用いるものとする．

## 3.2　アルゴリズムの概要

本節では，折り動作前の折り紙の状態と折り動作前後の折り紙の画像を用いることで折り線を検出し，折り動作後の折り紙の状態と行われた作業の認識を行う手法を提案する．本研究では，人間により行われた折り作業を観察し，どのような公理が適用可能であるかと，どのような折り紙の状態になったのかを認識することを目的とする．そのための入力情報

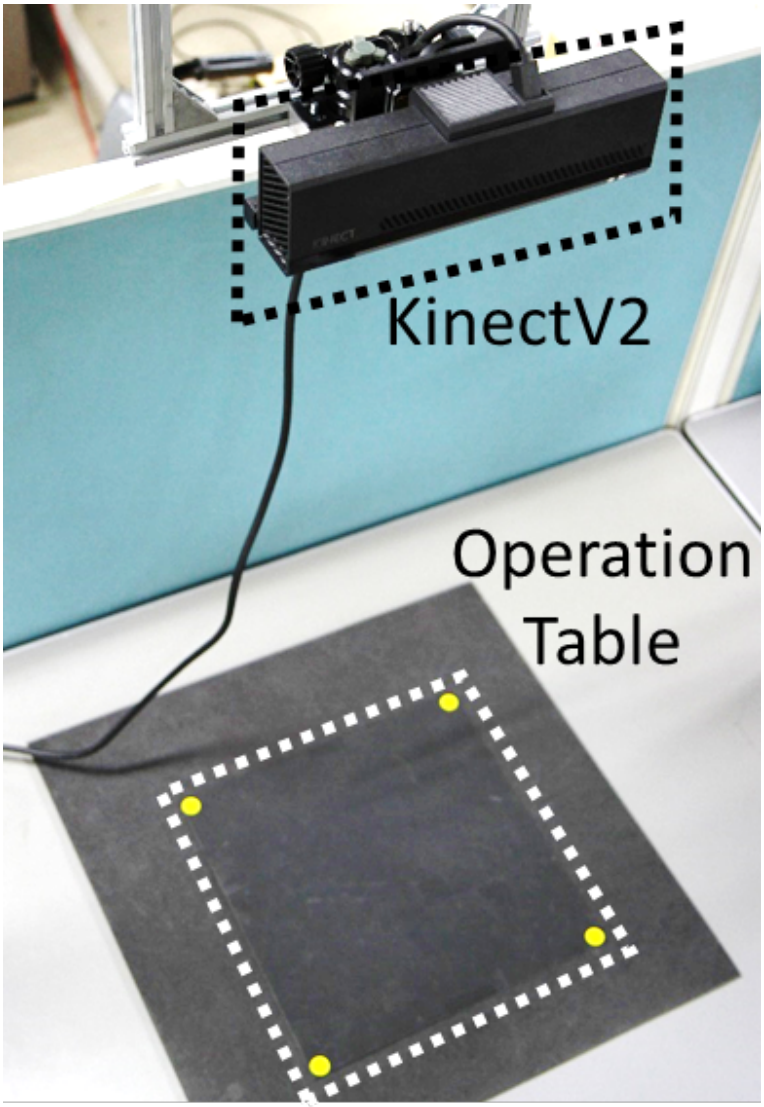



図 3.1: 撮影環境

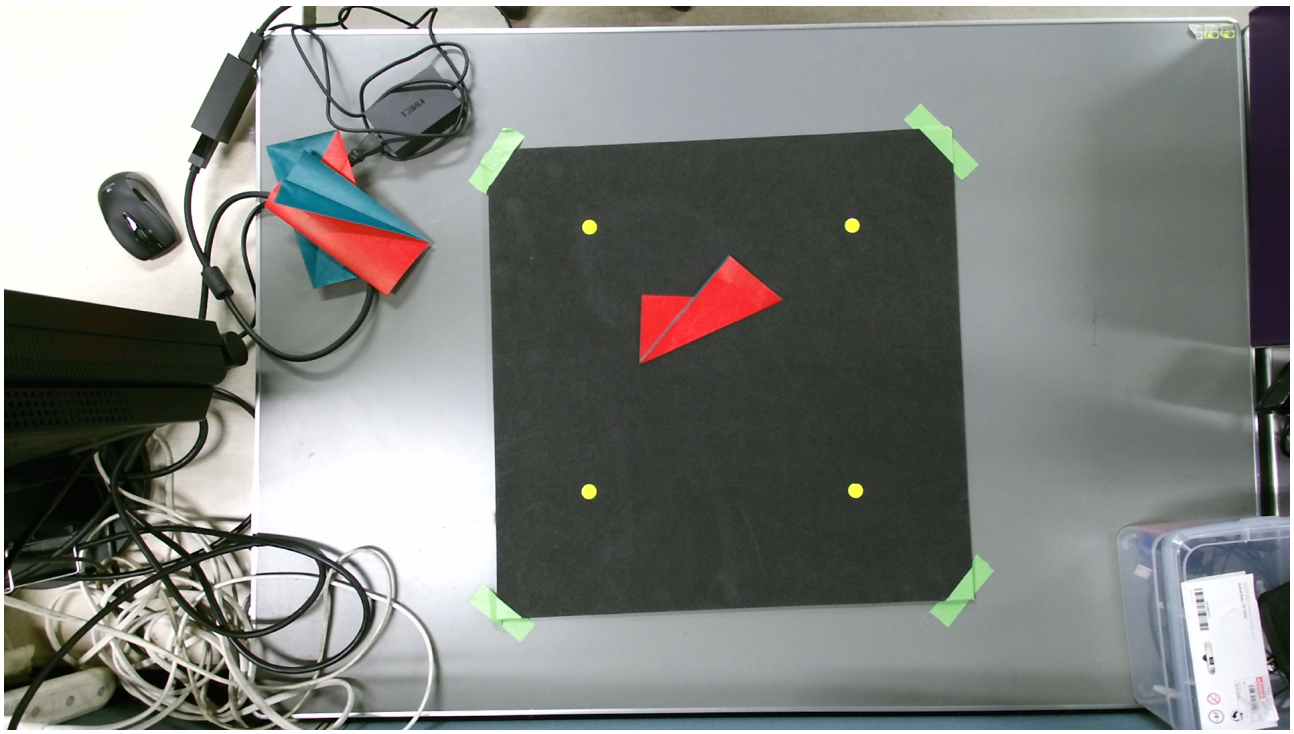


図 3.2: 切り抜き前の画像

図 3.3: 切り抜き後の画像

は，その折り動作が行われる前の折り紙の状態構造体 State（初期状態は既知として与えられる）と，折り動作前後の折り紙の色画像とする．出力情報としては，行われた作業の情報を Folding_Operation 構造体として，作業後の折り紙の状態の情報を State 構造体として得る．また，簡略化のため，以下の制約条件を設定した．

1. 折り紙の動作前後で折り紙のある頂点が移動および回転しないものとする
2. 折り紙は黒いマットの上におかれるものとし，マットには折り紙付近の領域を切り抜くことが可能となるように黄色のマーカを設置する．

これらの条件のもと，検討したアルゴリズムの概要を以下に記述し，図 3.4 に図示する．

Step1 折り線の位置を検出

Step2 折り動作後の状態を推定

Step3 適用可能な折り紙公理を列挙

以下ではこれらの概要について述べる．

### 3.2.1 折り線位置検出アルゴリズム

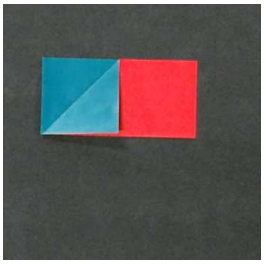
本節では折り作業で作成された折り線の位置を検出するアルゴリズムについて記述する．折り線の位置を検出するために，カメラを用いて折り紙を撮影し，折り動作の前後の画像の差分をとることで，折り線の位置を推定する手法を提案する．そのためにまず，折り方によりどのように折り線が見えるのかを 2 回折りまでに折り紙公理により記述可能な全ての折り作品に対して調査したところ，次の 3 つのパターンがあることが分かった．

1. パターン 1

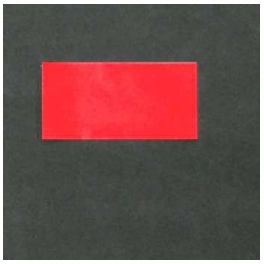
   折り紙の輪郭が変化し，かつ上から見た場合に初期状態の表面の色のみ見える折り方 (例：図 3.5(a))．

   折り紙が折り線で分断された片側全てを折り返すように折られるため，折り線は折り動作後の状態の外周上に存在する．

入力

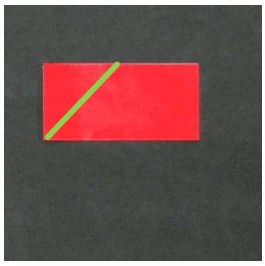


折り動作後画像

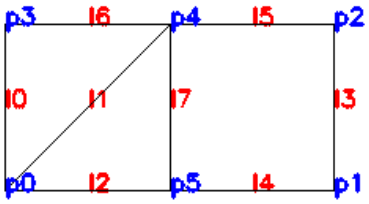


折り動作前画像

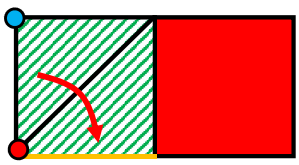
Step1:折り線を検出



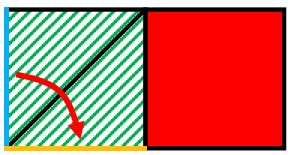
Step2:折り動作後の状態を推定




Step3:適用可能な公理を列挙



公理5



公理3

図 3.4: 折れ線検出に基づく折り操作の認識手法

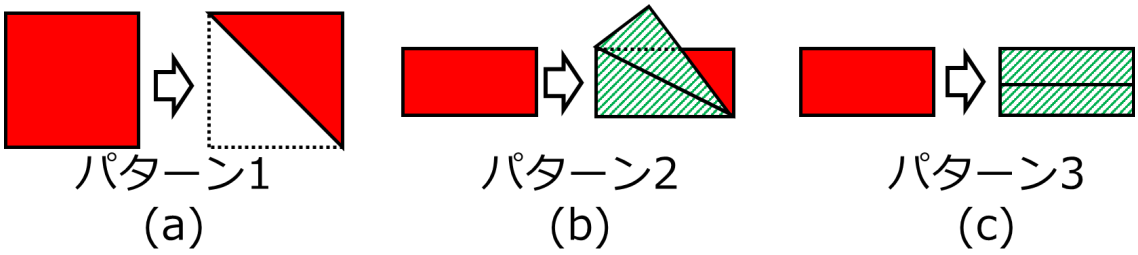



図 3.5: 折り方のパターンの例

2. パターン 2

   折り紙の輪郭が変化し，かつ上から見た場合に初期状態の両面の色が見える折り方 (例：図 3.5(b))．

   折り紙が一部めくられたように折られる．この際，めくられて新たな色が見えた部分を折り線で分断されるとちょうど二分されることになる．そのため，新たに見えた色の部分を 2 等分するような折り線が作られる．

3. パターン 3

   折り紙の輪郭が変化しない折り方（例：図 3.5(c))．

   折り線の現れ方はパターン 2 と同様である．

これら 3 つのパターンは，色の変化の仕方からパターン 1 を色の変化しない折り方，パターン 2，3 を色の変化する折り方とすることができる．色の変化しない場合の折り線位置推定手法として図 3.6 に示すフローの手法を検討した．また，色の変化する場合の折り線位置推定手法として図 3.7 に示すフローの手法を検討した．

### 3.2.2 状態推定アルゴリズム

状態推定アルゴリズムでは折り紙の動作前の状態構造体と折り線の位置，折り動作前後の特徴を入力として与えることで，折り動作が行われた後の折り紙の状態構造体を生成する手法を提案する．以下に今回検討した手法の流れを記述し，図 3.8 に図示する．

1. 折り紙の位置推定

   ICP アルゴリズムを用いて折り紙の位置を推定する．

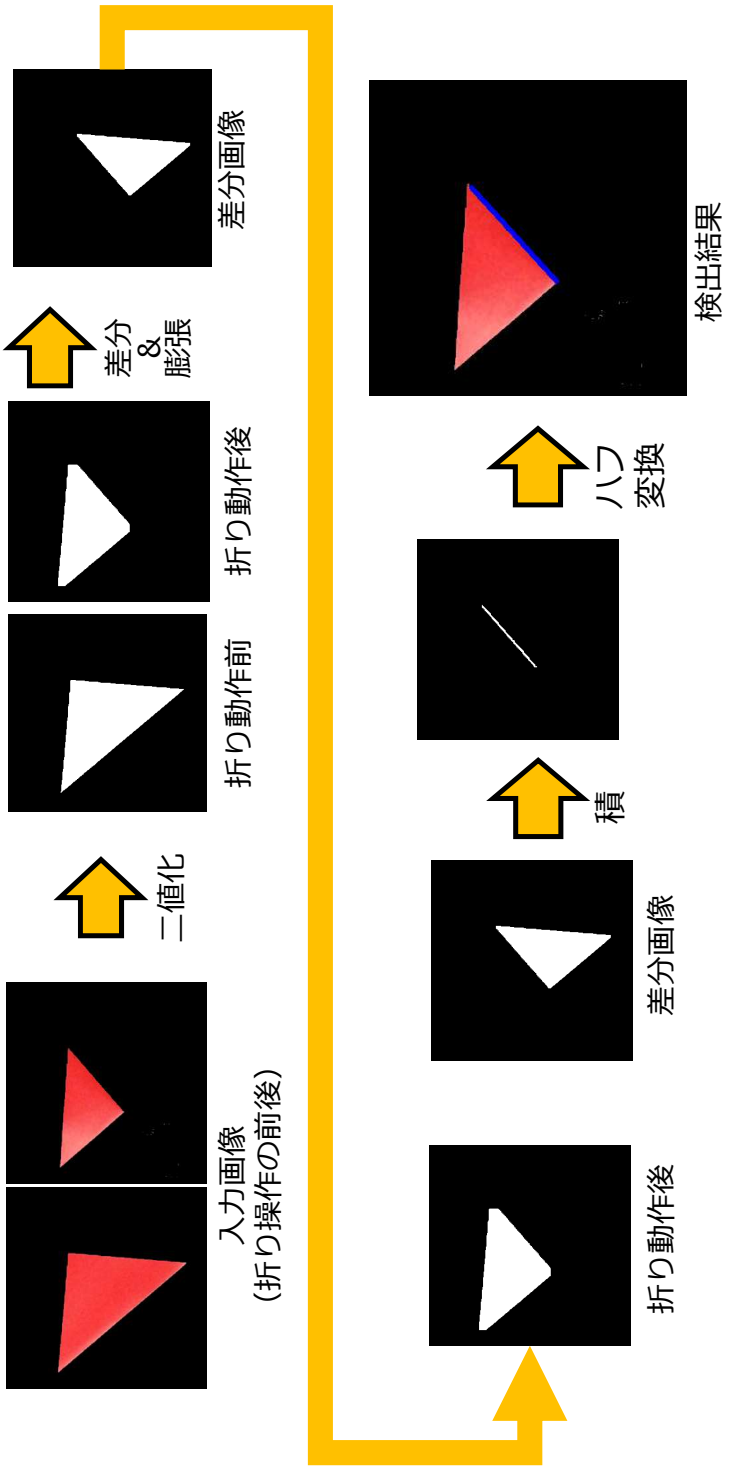



図 3.6: 色変化がない場合の折り線検出

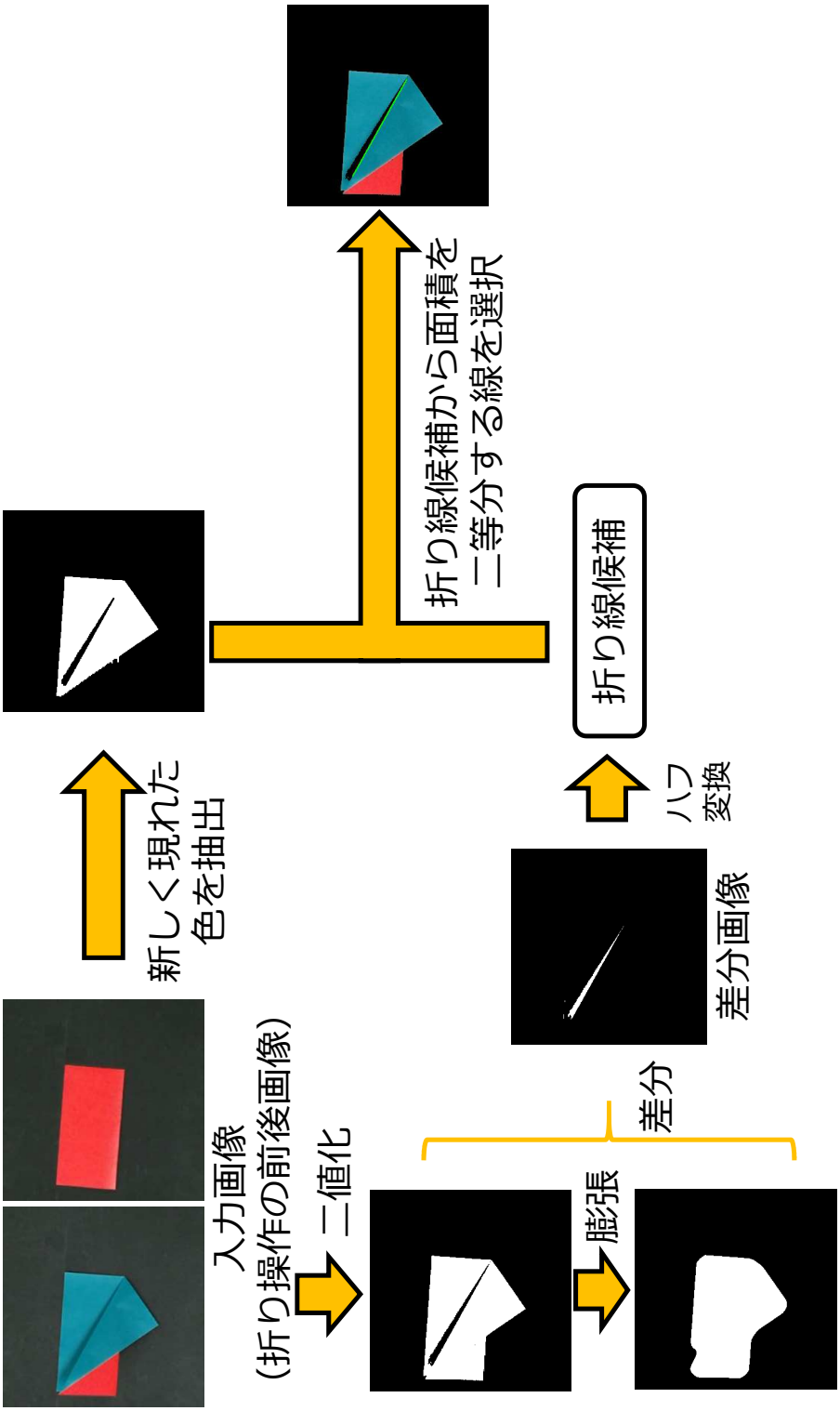



図 3.7: 色変化がある場合の折り線検出

2. 折り線により生成可能な状態の列挙

   折り線により折り返すことが可能なすべての状態を列挙する．

3. 状態候補の絞り込み

   列挙された状態を折り動作後の折り紙の特徴と比較することで，候補を絞り込む．

上記アルゴリズムでは，折り紙が折り線によって変化可能な状態をすべて列挙する．折り紙が折り線によって折り返される際に検討しなければならない問題として，どの面がどのように折り返されるかということがある．例えば単純に初期状態の正方形の状態で折り線を与えた際には図 3.9 のように折り線から見てどちらに折り返すかというだけで折り紙の状態は違うものとなる．また， 図 2.3 の (5) と（6）のように，折り線は同じであるがどの面を折り返すかの違いでも折り紙の状態は違うものとなる．また，第 2.2.2 節で検討した折り紙の状態構造体は一時的に重なった構造になってはいても 1 つの平面として記述しなければならない制約がある．そのため，推定された折り紙の状態を 1 つの面として記述することも必要となる．

### 3.2.3 動作推定アルゴリズム

第 3.3.2 節で得られた折り紙の状態に対し，入力された折り紙の状態 State と比較することでどの折り紙公理が適用されたかを計算し，行われた作業を構造体として記述するアルゴリズムを検討する．

アルゴリズムでは各公理ごとにその公理が適用可能であるかを調べ，適用可能であれば折り紙公理に必要なパラメータを Axiom_Info 構造体として出力するものを目的とする．

特に，今回は 2 回折りまでに折り紙公理で記述可能な折り線には公理 5 までしか出てこないため，公理 5 までのみ検討することとした．また，第 3.3.1 節で検出された折り線の位置は，あくまで画像認識により得られたものであり，実際の折り線の位置からずれたものである場合が多い．そのため，適用可能な折り紙公理から折り線を再度生成することで理想的な折り線の位置も検出する．

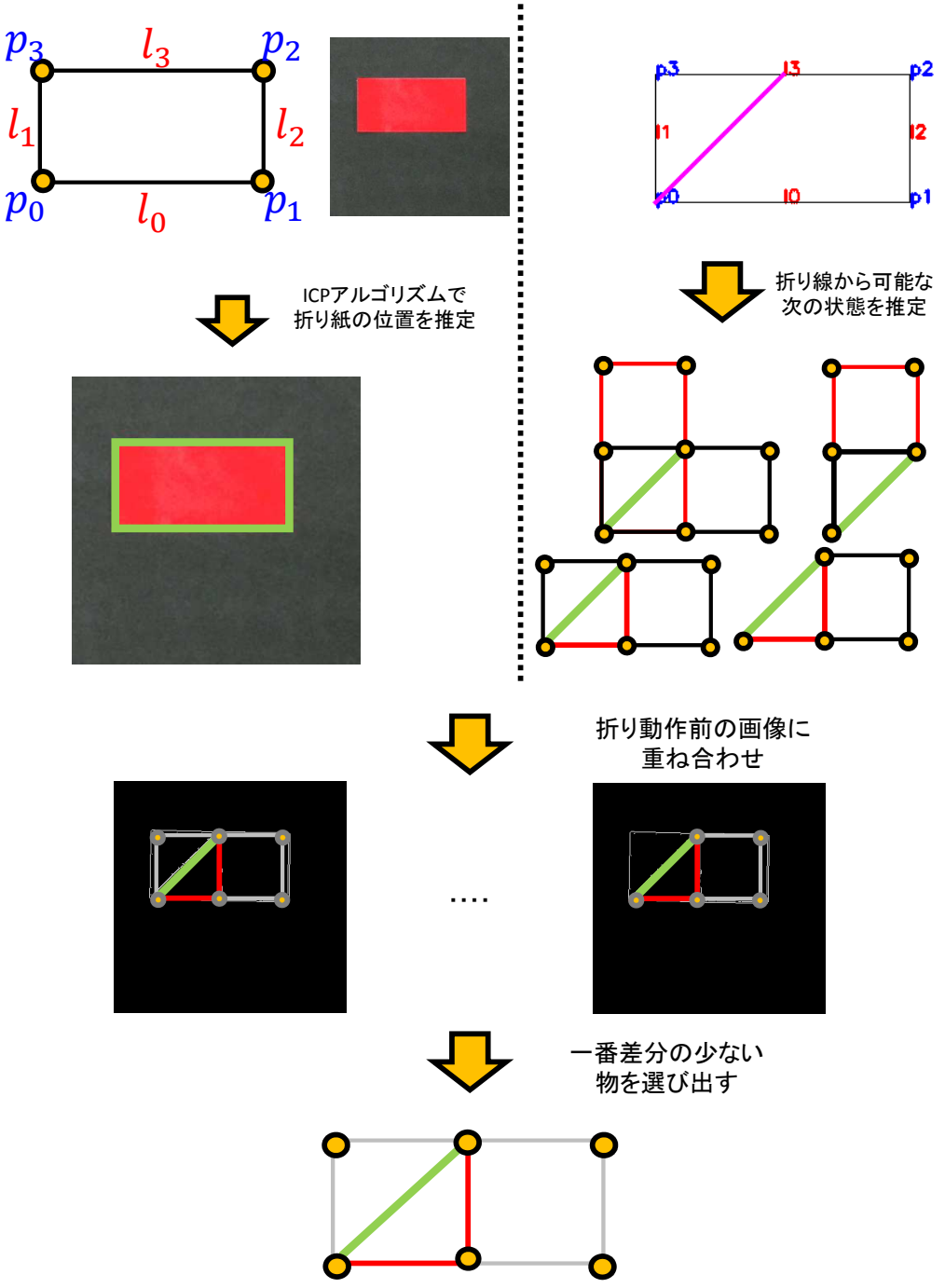



図 3.8: 状態推定アルゴリズム

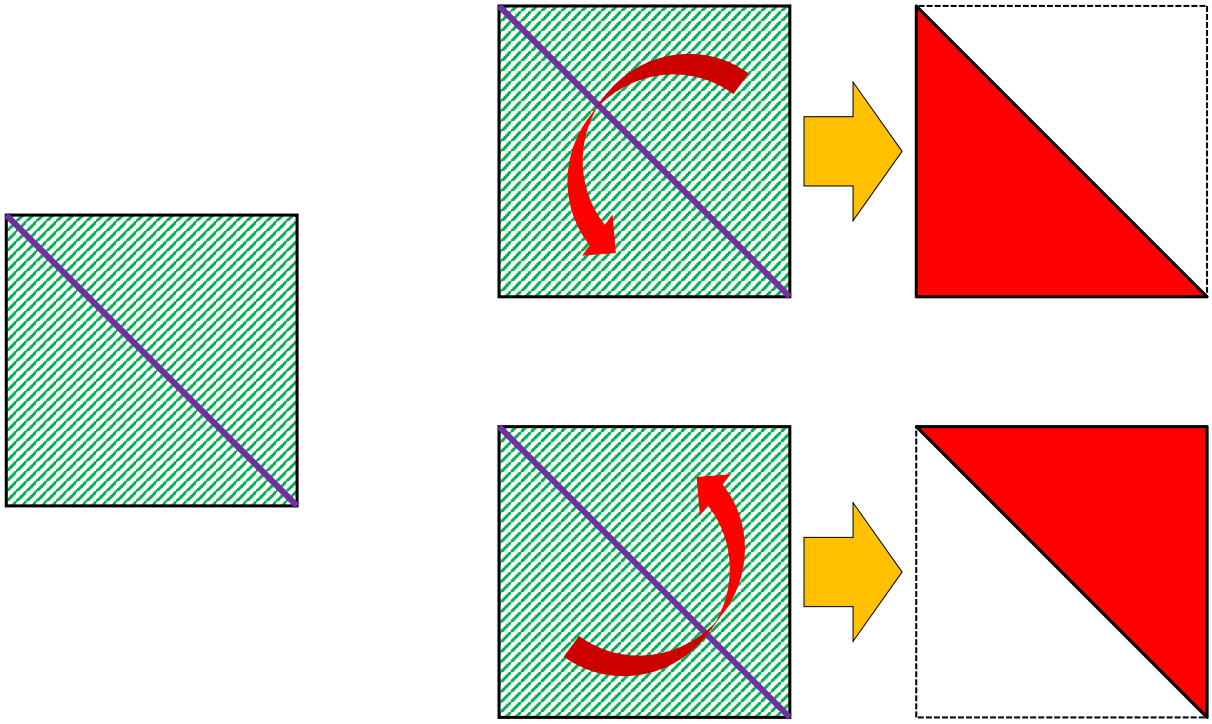


図 3.9: 折り操作を行う方向による状態の違い

## 3.3　アルゴリズムの詳細

本節では前節で記述したアルゴリズムの詳細を記述する.

### 3.3.1　折り線検出アルゴリズム

以下に，前述した二つのパターンの分け方，色変化のない場合の折り線検出アルゴリズム，色変化のある場合の折り線検出アルゴリズムを示す.

#### パターン分類後に使用する折り線検出手法

折り紙の折り方を図 3.5 に示す 3 つのパターンに分ける手法を提案した．そのために，折り動作前の画像 A と折り動作後の画像 B に対し，それぞれの 2 値画像である $A_b$ と $B_b$ の白い部分の面積を比較し,「増加した」,「減少した」,「あまり変化がない」の 3 つのパターンに分けた．以下に，分けたパターンごとに使用する折り線の検出手法を記す.

1. 増加した場合

   色が折り紙上に何色あるかを観測し，色の数により以下に分岐する.

   (a) 1 色の場合 (パターン 1)

   色変化のない折り線検出を実施.

   (b) 2 色の場合 (パターン 2)

   色変化のある折り線検出を実施.

2. 減少した場合 (パターン 1)

   色変化のない折り線検出を実施.

3. 変化がない場合 (パターン 3)

   色変化のある折り線検出を実施.

色変化のない折り線検出手法

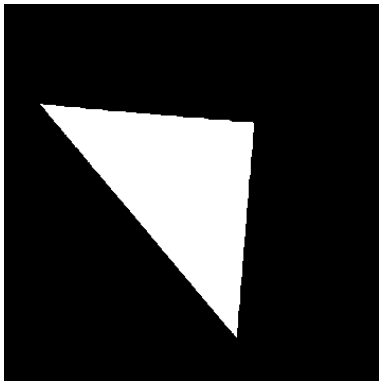
折り紙が輪郭の変化する形で折られたと仮定し，検出を行った．その手順を以下に記し，その処理過程の画像を図 3.10 に示す．また，以下では，折り紙の折り動作前の画像を A，折り動作後の画像を B とする．

1. A，B に対し輝度による 2 値化を行った画像 $A_b$，$B_b$ を作成
2. $A_b \bigcap \bar{B_b}$ となる画像 (C) を作成．
3. $B_b$ を膨張処理 ($B_{bd}$)．
4. C を膨張処理 ($C_d$)．
5. $B_{bd} \bigcap C_d$ となる画像 (D) を作成．
6. D に確率的ハフ変換を行い，得られた線分の中から一番長いものを折り線として抽出．

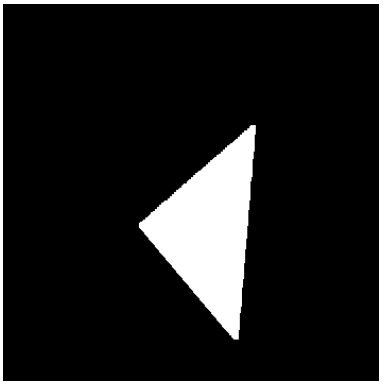
色変化のある折り線検出手法

折り紙が輪郭の変化しない形で折られたと仮定し，検出を行った．その手順を以下に記し，その処理過程の画像を図 3.11 に示す．以下では，折り紙の折り動作前の画像を A，折り動作後の画像を B とする．

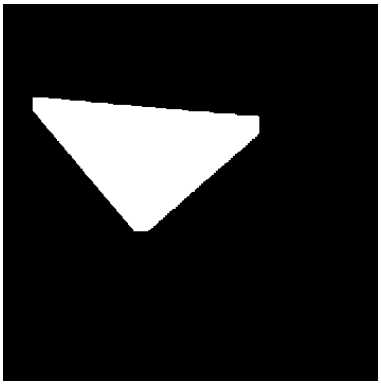
1. B に対し輝度による 2 値化を行った画像 $B_b$ を作成
2. $B_b$ に対し，膨張処理を適用した画像 ($B_d$) を作成
3. $B_b$に存在し $B_d$に存在しない部分の 画像 (E) を作成
4. E に確率的ハフ変換を行い直線を検出
5. A と B で，B にしかない色の部分となる 2 値画像 (F) を抽出．
6. 3) で得られた線分で F を分断した場合に，白い部分が一番均等に分断される線分を折り線とする．



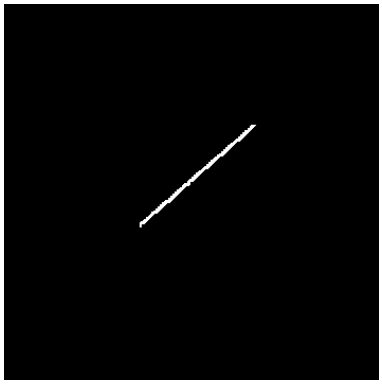
(a) $A_b$



(b) $B_b$



(c) C

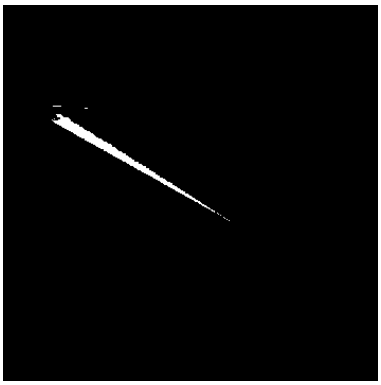


(d) D

図 3.10: 色変化のない場合の折り線検出手法

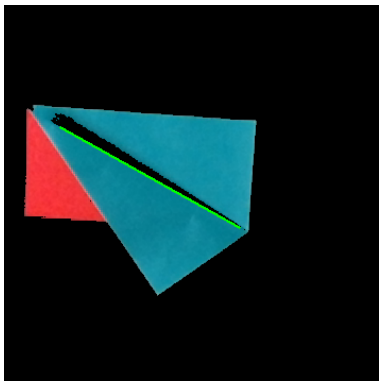
(a) $B_b$

(b) $B_d$



(c) E

(d) A



(e) B

(f) F

図 3.11: 色変化のある場合の折り線検出手法

### 3.3.2 状態推定アルゴリズム

前述した状態推定アルゴリズムで行われる各手法の詳細を記述する．

#### 折り紙位置の推定

本稿の折り紙の状態 State の折り紙の座標は折り紙座標系で記述されており，実際に取得した画像中のどこに存在するかはその都度探索しなければならない．そのため，本節では，折り紙の状態 State が画像上のどこに存在するかを ICP アルゴリズム [28] を用いて，折り紙の状態を画像上の折り紙の位置に移動するための回転行列と並進ベクトルを求める手法について述べる．回転行列と並進ベクトルを求めることで，折り紙の点が折り紙の画像上のどこに対応するのかを計算可能とする．以下にはその手法について記述する．

1. 折り紙の状態候補の State 構造体の辺の内外周の辺のみをつなぐように線を描画した 2 値画像 SV を作成する．
2. A に対し輝度による 2 値化を行う ($A_b$)
3. $A_b$ に対し，Canny アルゴリズムにより外周のエッジを取得する ($A_o$ )
4. SV 及び $A_o$ の白い部分のピクセルを画像上の座標系で記述される点群に変換する（$SV_p$, $A_{o\_p}$）．
5. $SV_p$ をデータ行列（位置合わせのために移動される点群），$A_{o\_p}$（位置合わせされる点群）をモデル行列として ICP アルゴリズムを適用し，回転行列と並進ベクトルを求める．

ICP アルゴリズムでは問題点として最近傍点探索の際の計算時間があげられている．計算時間に関しては次の対処を行っている．最近傍点探索の問題点は 2 つを考慮した．そのうちの 1 つである点群の数による問題に関しては，作成する点群を乱数により減らすことで対処している．もう 1 つの繰り返し回数であるが，繰り返し回数は実験により最低限検出可能な

回数を決定した．以降の手法ではこの手法により折り紙の位置がわかっているものとして処理を行う．

### 折り線により生成可能な状態候補の列挙

折り紙の状態と折り線により生成可能な状態候補の列挙するアルゴリズムを以下では提案する．

折り紙の状態を折り線により折り返すためには，どの面が折り返されたかを認識する必要がある．またそれだけでなく，図 3.12 の場合のように折り紙が 2 枚重なっているような場合には同じ折り線で同じ面を折り返す際に 2 つの場合が存在してくる．これらのことから，折り返す面を認識し，2 つのパターンで折り返すようなアルゴリズムが必要と考えた．折り返す面の探索方法は折り線を直線として扱うか線分として扱うかで 2 つのパターンを考えた．以下にそれら手法を記述する．

1. 折り直線による折り返し
   (a) 折り紙上に折り線の重なる点を分断点 $V_{dev}$ として保存する．
   (b) 折り紙上の全ての頂点に対して折り線で分断し，頂点の ID を保存した点群 $V_r$,$V_l$ を作成
   (c) $V_r$ の ID を持つ頂点を折り線に対し線対称となる点に移動した State 構造体を候補群（Candidate_States）に追加する．
   (d) 移動した頂点と辺，分断点 $V_{dev}$ の情報を，候補群（Candidate_Operations）に追加する．
   (e) 上記の 2 つの $V_r$ に対して行った処理を $V_l$ に対しても行う．
   (f) $V_r$ の ID を持つ頂点に対し，同じ位置に新たな ID を保持する頂点を追加した State 構造体の $State_o$ を生成．
   (g) $State_o$ の経路候補の ID を持つ頂点を折り線に対し線対称となる点に移動した State 構造体を候補群（Candidate_States）に追加する．

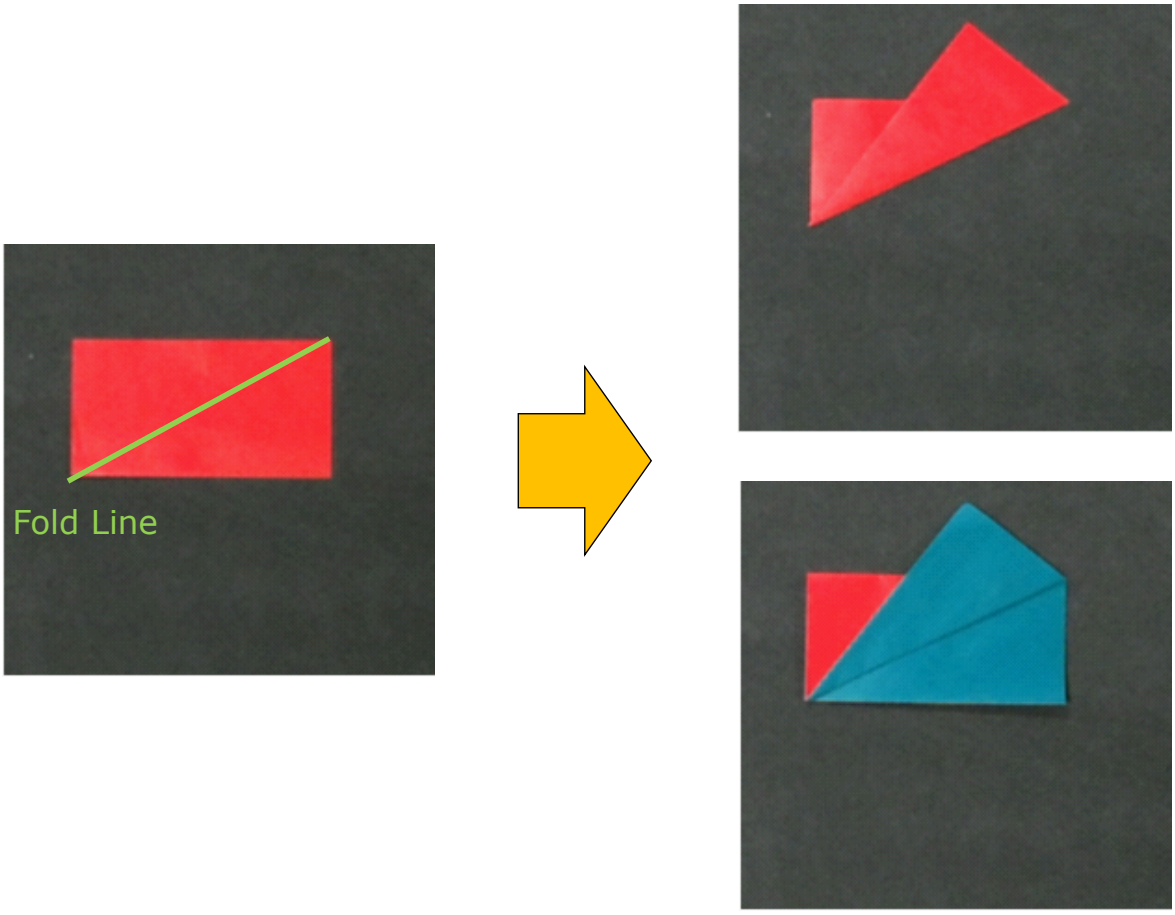



図 3.12: 2 枚の紙が重なっている場合

(h) 移動（折り返し）した頂点と新たに追加した辺，分断点の情報を，候補群（Candidate_Operations）に追加する．

(i) 上記の 3 つの $V_r$ に対して行った処理を $V_l$ に対しても行う．

2. 折り線分による折り返し

(a) 折り紙上に折り線の重なる点を分断点 $V_{dev}$ として保存する．

(b) $V_{dev}$ の重なる線を $V_{dev}$ で分断し，2 つの線とする

(c) 折り紙上の全ての頂点に対して折り線で分断し，頂点の ID を保存した点群 $V_r$,$V_l$ を作成

(d) 分断点 $V_{dev}$ の 1 つを開始点として別の分断点 $V_{dev}$ へ到達可能な経路をすべて列挙する．以下に 1 つの経路当たりの検出アルゴリズムを記述する．

i. 分断点の 1 つを開始点としてキュー Que に保存する．

ii. 目標点 S として分断点の開始点ではない点を設定

iii. キューの 1 番最後を取り出す $Q_{last}$

iv. 現在地として Now に $Q_{last}$ の値をコピー

v. Now が目標点であればその時点での Route を経路の 1 つとして保存

vi. Now が行き止まりか目標点であれば，以下の処理をする

A. キュー Que の 1 番最後をコピーし，$Q_{last}$ に保存する

B. State の辺の頂点 ID の内 $Q_{last}$ の ID を含むものが 3 つある場合は終了

C. State の辺の頂点 ID の内 $Q_{last}$ の ID を含むものが 2 つ以下の場合はキュー Que の 1 番最後を消去し A に戻る

vii. Now が行き止まりでもゴールでもない場合，State の辺の頂点 ID の内 $Q_{last}$ を含む辺をすべて探索し，$Q_{last}$ ではないほうの頂点の ID を，Route に存在しない物以外全てキュー Que の 1 番後ろに追加する．

viii. キューの数が 0 個なら終了

ix. キューの数が 0 個でないなら iii に戻る

(e) 全ての保存した経路候補に対し，以下の処理を行う．

i. 経路候補の ID を持つ頂点を折り線に対し線対称となる点に移動した State 構造体を候補群（Candidate_States）に追加する．

ii. 移動（折り返し）した頂点と辺，分断点の情報を，候補群（Candidate_Operations）に追加する．

i. 経路候補の ID を持つ頂点に対し，同じ位置に新たな ID を保持する頂点を追加した State 構造体の $\text{State}_o$ を生成．

ii. $\text{State}_o$ の経路候補の ID を持つ頂点を折り線に対し線対称となる点に移動した State 構造体を候補群（Candidate_States）に追加する．

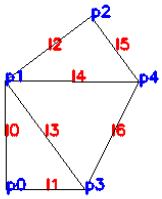
iii. 移動（折り返し）した頂点と新たに追加した辺，分断点の情報を，候補群（Candidate_Operations）に追加する．

以降の処理においてこの処理で生成した Candidate_States 及び Candidate_Operations を用いる．上記のアルゴリズムであるが，生成される候補は最低 2 つ（折り線から見て左右両方で折り返すため）存在するため，絞り込みのためのアルゴリズムが必要である．図 3.13 に生成された折り紙の状態例を示す．以下に示すのは第 2.3 節の (2) から (12) が作られる過程で列挙された結果の一部を可視化したものである．

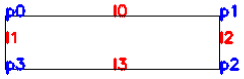
#### 状態候補群の絞り込み

折り線により生成可能な状態候補の列挙で生成された折り紙の状態候補群（Candidate_States）は必ず 2 つ以上存在するため，その中から正しいものを選ぶ必要がある．そこで，正しいものを選ぶために折り動作後の折り紙の形状に最も近いものを選び出すアルゴリズムを提案する．

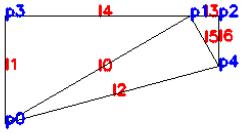
1. 以下の手法により折り動作後の折り紙の画像に対し，特徴を抽出する．この時の処理過程を図 3.14 に示す．



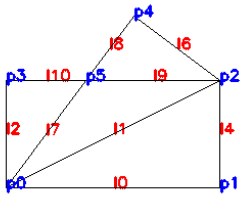

(a) 1



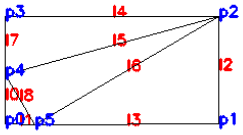

(b) 2




(c) 3




(d) 4




(e) 5

図 3.13: 列挙された結果の一部

(a) カメラからマーカにより切り抜かれた折り紙の画像の取得 (A)

(b) A に対し輝度による 2 値化を行う ($A_b$)

(c) $A_b$ をマスク画像として，A を切り抜く ( $A_c$ )

(d) $A_c$ に先鋭化フィルタを適用する ($A_{sf}$)

(e) $A_{sf}$ にバイラテラルフィルタを適用する ($A_{bf}$ )

(f) $A_g$ をグレースケール化する ($A_g$ )

(g) $A_g$ に対し，Canny アルゴリズムによりエッジを取得する ($A_c$ )

2. 以下の手法によりすべての候補画像に対し，誤差値を計算する．

(a) 折り紙の状態候補の State 構造体の辺をつなぐように線を描画した 2 値画像 SV を作成する

(b) $B_c$ に対し膨張処理を適用した 2 値画像 $B_{dil}$ を作成

(c) $B_{dil} \bigcap$SV となる部分を SV から排除した 2 値画像 $SV_{extract}$ を作成する

(d) $SV_{extract}$ の白い部分の面積を SV の白い部分の面積で割った値 param1 を作成

(e) floodFill アルゴリズムを $B_{dil}$ に適用し，内部を埋めた 2 値画像 $B_{fill}$ を作成する．

(f) floodFill アルゴリズムを SV に適用し，内部を埋めた 2 値画像 $SV_{fill}$ を作成する．

(g) $B_{fill}$ に存在し $SV_{fill}$ に存在しない部分の画像 $Diff_1$ を作成する．

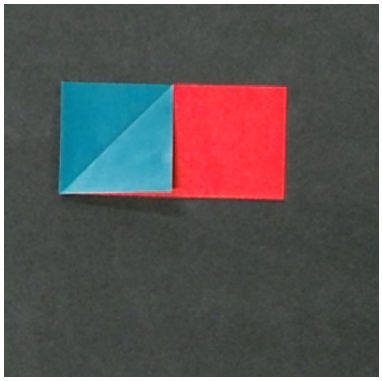
(h) $SV_{fill}$ に存在し $B_{fill}$ に存在しない部分の画像 $Diff_2$ を作成する．

(i) $Diff_1$ の白い部分の面積を SV の白い部分の面積で割った値 param2 を作成

(j) $Diff_2$ の白い部分の面積を SV の白い部分の面積で割った値 param3 を作成

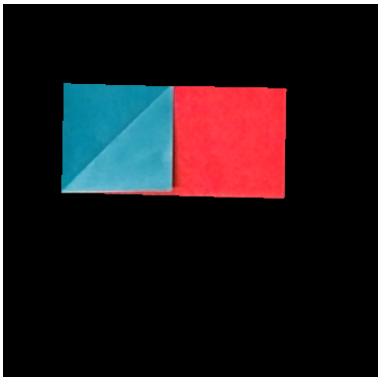
(k) param1 と param2 と param3 を乗算した値を誤差値として出力する．

3. 得られた各候補の誤差値の内，一番値の小さいものを折り動作後の折り紙の画像に近い画像として出力する．

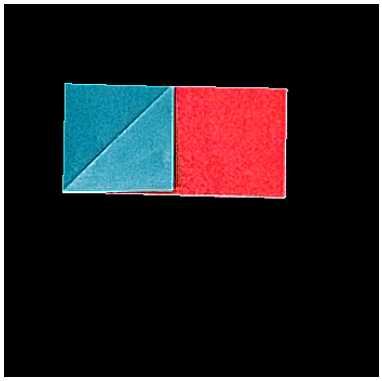


(a) A

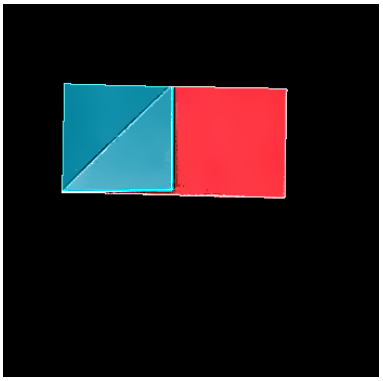
(b) $A_b$



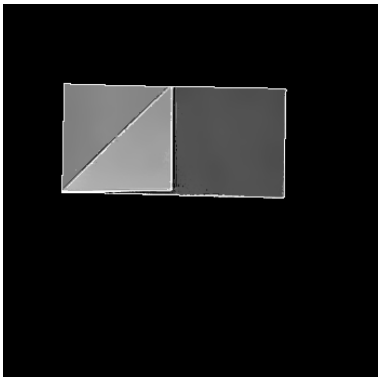
(c) $A_c$



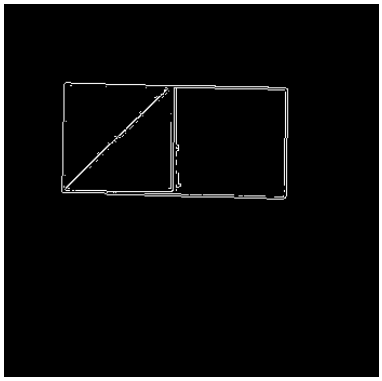
(d) $A_{sf}$



(e) $A_{bf}$



(f) $A_g$



(g) $A_c$

図 3.14: 特徴の抽出過程

### 重なり構造の排除アルゴリズム

状態候補群の絞り込みで決定された折り紙の候補 $State_{extract}$ に対し，重なり構造を排除し，一面として記述するためのアルゴリズムを示す.

1. $State_{extract}$ の全ての頂点に対し，以下の処理を行う
   (a) $State_{extract}$ の内，1 つの頂点を $P_s$ とする.
   (b) $P_s$ が，発見済み点群 $F_{group}$ に存在しない場合，$F_{group}$ と新頂点群 $NewV_{group}$ に追加する.
   (c) $P_s$ が，発見済み点群 $F_{group}$ に存在する場合，1 に戻る.
   (d) $P_s$ から近い距離にある点 $State_{extract}$ の中から探索し，存在すれば $F_{group}$ に追加する.
   (e) すべての頂点を探索し終えれば終了
2. $NewV_{group}$ の頂点に対して全ての組を作成し，それらを辺候補群 $CandidateE_{group}$ とする.（逆順の組み合わせになるものは排除する）
3. $CandidateE_{group}$ の各辺に対し，$State_{extract}$ 上に近い辺があるかを計算し，存在する辺のみを新たな候補群 $CandidateE_{extract}$ に保存する．また，その際に見つかった近い辺は全て old_id として保存する.
4. $NewV_{group}$ の各頂点に対し，$State_{extract}$ 上に距離の近い頂点が存在するかを探索し，見つかった点の ID は old_id にそれぞれの頂点ごとに保存する.
5. 新しく State 構造体で $State_{new}$ を作成する.
6. $State_{new}$ の頂点群 vertices に $NewV_{group}$ を代入
7. $State_{new}$ の辺群 $edges$ に $CandidateE_{extract}$ を代入
8. $State_{new}$ の origami_first_edge_size に $State_{extract}$ の origami_first_edge_size を代入

9. $State_{new}$ に対し，$State_{new}$ 上の線全ての交点を計算し，$State_{new}$ の点の近くにない交点を計算する．

10. $State_{new}$ 上で $V_{dev}$ の重なる線を $V_{dev}$ で分断し，2 つの線とする．

11. $State_{new}$ を結果として出力

### 3.3.3 動作推定アルゴリズム

前述した，適用可能な折り紙公理の推定アルゴリズムの詳細を記述する．ここでは，各公理ごとのパラメータの推定手法をまず述べ，その後に，各公理ごとの折り線の計算方法を述べる

#### 適用可能な公理の推定

1. 公理 1

   公理 1 とは，与えられた 2 点を通る線で折る．これは折り線が折り動作前の折り紙の頂点のうち 2 つ以上が折り線に重なっている場合にこの公理が適用可能であると言える．よって以下では折り線上に折り動作前の折り紙の頂点がいくつ存在するかを調べ，存在した全ての組みを適用可能な公理として出力するアルゴリズムを示す．

   (a) $State_{before}$ の全ての頂点に対し，折り線に下した垂線の距離が短いものを $P_{near\_orisen}$ に保存する

   (b) $P_{near\_orisen}$ の全ての組を作成し，その組全てに対し折り紙公理構造体 Axiom_Info を作成し，折り操作構造体 Folding_Operation の axiom_infos に追加する．

2. 公理 2

   公理 2 では，与えられた 2 点を重ねて，2 点間の垂直 2 等分線で折る．これを検出するために，$State_{new}$ の頂点構造体にある old_id を用いた．これは，この点が前回どの点

であったかというものであり，これが2つあるということは，この点は以前に対して2つの点が重なってできているものということになる．

(a) $\mathrm{State_{new}}$ の各頂点構造体にある old_id が 2 つ以上存在する場合それら全ての組 $\mathrm{V_{pairs}}$ を作成する．

(b) $\mathrm{V_{pairs}}$ の組に対し，その組の情報 2 つを公理の情報として Axiom_Info に保存し，折り操作構造体 Folding_Operation の axiom_infos に追加する．

3. 公理 3

公理 3 では，2 線 $\mathrm{l_1}, \mathrm{l_2}$ から折り線を定義する．これが適用可能であるか判定するために，$\mathrm{State_{new}}$ の辺構造体にある old_id を用いた．処理方法は公理 2 とほぼ同じである．いかにそのフローを記述する．

(a) $\mathrm{State_{new}}$ の各辺構造体にある old_id が 2 つ以上存在する場合それら全ての組 $\mathrm{E_{pairs}}$ を作成する．

(b) $\mathrm{E_{pairs}}$ の組に対し，その組の情報 2 つを公理の情報として Axiom_Info に保存し，折り操作構造体 Folding_Operation の axiom_infos に追加する．

4. 公理 4

公理 4 は， 1 点 $\mathrm{p_1}$，1 線 $\mathrm{l_1}$, から折り線を定義する．これが適用可能であるか判定するために，次の 2 つの条件が成立するかを調べた．1 つ目は，折り線上に頂点が 1 つ以上存在するか．2 つ目は，折り線に対し垂直な辺が存在するかである．以下ではこれらを調べる処理について記述する．

(a) 公理 1 と同じ方法で折り線上にある頂点を全て保存する $\mathrm{P_{near\_orisen}}$.

(b) $\mathrm{State_{new}}$ の各辺に対し，折り線からの角度を計算し，その角度が 90 度に近いものを全て保存する（$\mathrm{Edges_{vertical}}$）．

(c) $\mathrm{P_{near\_orisen}}$ と $\mathrm{Edges_{vertical}}$ で可能な全ての組み合わせを代入した Axiom_Info を作成し，折り操作構造体 Folding_Operation の axiom_infos に追加する．

5. 公理5

   公理5は，2点$p_1, p_2$，1線$l_1$から折り線を定義する．これが適用可能であるか判定するために，次の2つの条件が成立するかを調べるアルゴリズムを作成した．1つ目の条件は，折り線上に頂点が1つ以上存在するか．2つ目は，折り返した頂点が辺上に重なったものが存在するかである．以下にはそれらを調べるアルゴリズムについて記述する．

   (a) 公理1と同じ方法で折り線上にある頂点を全て保存する$P_{near_orisen}$.

   (b) $State_{new}$の各辺に対し，Candidate_Operationsの折り返した点が公理1と同じように辺上に重なっているものがあるか探索し，存在すればその辺と頂点の全てを1つの組として保存する$VE_{near_line}$.

   (c) $P_{near_orisen}$と$VE_{near_line}$で可能な全ての組み合わせを代入したAxiom_Infoを作成し，折り操作構造体Folding_Operationのaxiom_infosに追加する．

### 公理による折り線

動作推定アルゴリズムにより適用された公理と，公理で折り線を生成するために必要となるパラメータが計算された．本節では適用された公理で折り線の位置を計算する方法について記述する．

本節で検討する折り線は，2つの点として記述されるが，折り紙公理では折り線は直線として扱われるため，直線として扱う．

1. 公理1

   公理1では2点$p_1, p_2$をつなぐ折り線を定義する．

2. 公理2

   公理2により作られる折り線は，与えられた2点$p_1, p_2$の垂直二等分線により生成される．そのため，Stateの全ての頂点の組み合わせに対して折り線を列挙する．この時，頂点の各組合せに対し，以下のような処理を行うことで折り線を出力する．

   (a) 2つの点の組み合わせ$p_1$，$p_2$の中心点$p_{center}$を求める．

(b) $p_{center}$ を中心に，$p_1$ を 90 度回転した点 $p_{turned}$ を求める．

(c) 折り線の頂点に $p_{center}$ と $p_{turned}$ をとする．

3. 公理 3

公理 3 により作られる折り線は，2 線 $l_1, l_2$ の 2 等分線により生成される．2 等分線の生成には 2 線のベクトルを生成し，その和のベクトルと和のベクトルに直行するベクトルの 2 つが可能であり，それら 2 つの両方を出力し，その中から第 3.3.1 節で検出された折り線に近いほうを選ぶ．以下のような処理を行うことで折り線を出力する．

(a) 2 線 $l_1, l_2$ のベクトル $Vl_1, Vl_2$ を求める．

(b) ベクトル $Vl_1, Vl_2$ の和 $V_{sum}$ を求める．

(c) 2 線 $l_1, l_2$ の交点 $Cross_V$ を求める．

(d) $Cross_V$ を開始点として $V_{sum}$ 分進んだ点を計算する $P_{candidate1}$．

(e) $Cross_V$ を中心として $P_{candidate1}$ を 90 度回転した点 $P_{candidate2}$ を求める．

(f) $p_{center}$ を中心に，$p_1$ を 90 度回転した点 $p_{turned}$ を求める．

(g) 折り線の頂点に $p_{center}$ と $p_{turned}$ をとする．

4. 公理 4

公理 4 は与えられた 1 点 p と 1 本の，l に対し，点を通るように線を自分自身の上に重ねて，与えられた点を通るように与えられた線の垂直 2 等分線で折る．折り線は，p と p から l に下した垂線との交点 $p_{vertical\_cross}$ で得られる．

5. 公理 5

公理 5 は与えられた 2 点 $p_1, p_2$ と 1 本の線 l に対し，1 方の点 $p_2$ が 1 方の線 l に乗るように他方の点 $p_1$ を通る線で折る．折り線の計算のためには $p_2$ を中心とし，$p_1$ と $p_2$ の間の距離の半径を持つ円と，l の交点を求める．そのため，2 つの候補が出てくるが，その中から第 3.3.1 節で検出された折り線に近いほうを選ぶ．

(a) $p_1$ と $p_2$ の間の距離を計算し，円の半径 r とする．

(b) $p_2$ を中心とし r の半径を持つ円と l の交点 $p_{cross1}$，$p_{cross2}$ を求める．この時，交点が 1 つであれば，その交点と $p_1$ を結果として出力する．

(c) $p_1$ を通り $p_{cross1}$ または $p_{cross2}$ を通る折り線の内，第 3.3.1 節で検出された折り線に近い交点 $p_{nearcross}$ を求める

(d) 折り線の頂点を $p_1$ と $p_{nearcross}$ とする．

## 3.4 実験

実験では，本節のアルゴリズムについて 2 回折りまでの折り紙に対し，1 回ずつ実験を行い，本節のアルゴリズムが正常に記述できるかを示す．実験対象となる折り紙は第 2.3 節で列挙された 2 回折りまでの折り紙を対象とする．実験で検討する内容は，折り紙の状態が正しく記述されているか，折り紙操作が正しく記述されているかとする．また，処理ごとに必要になった時間も検討する．

### 3.4.1 実験結果

本節では，第 2.3 節で列挙した 2 回折りまでに折り紙公理で表現可能な全ての折り作品について作業が推定可能であるかを実験する．

実験結果は，入力された画像と，それにより検出された折り線，行われた作業を 1 つの折り作業ごとに記述する．作業記述の際には，適用された公理の番号と，第 2.2.3 節で定義された各公理に対応する関数の引数の順番で記述される．また，ここで記述されるのは ID のみであり，その ID は折り線の表示される折り紙の ID に対応する．今回，初期値として図 3.15 のように可視化する折り紙の状態を与えた．可視化する際，$(p_0)$ を座標系原点として記述した．これら結果では，全てにおいて正しく谷折操作の推定及び折り線の位置の推定が行うことが本アルゴリズムで可能なことが確かめられた．

以降，これを初期値として 2 回折りまでの折り紙の作業を推定する．以降，図 4.8 から図 3.31 までに動作推定結果の図を，表 4.3 から，表 3.16 に操作推定結果の表を示す．

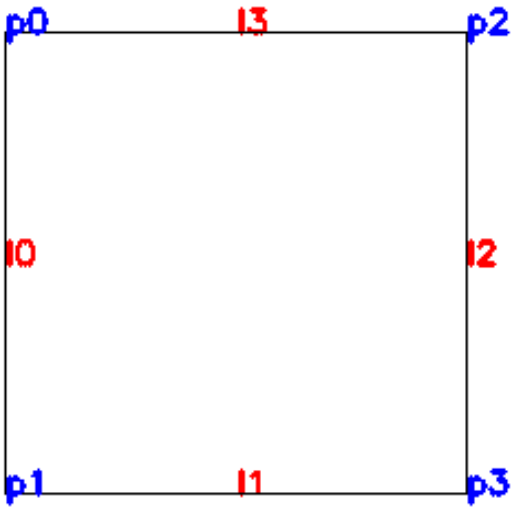
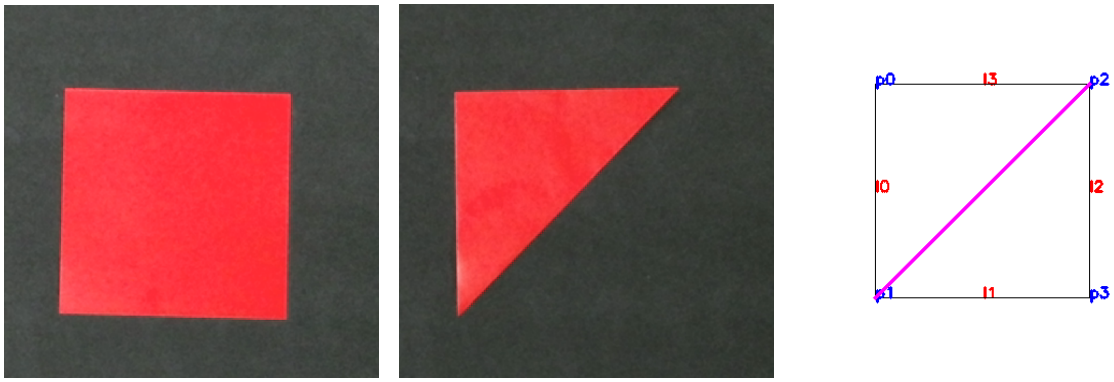



図 3.15: 初期値として与えた折り紙の状態

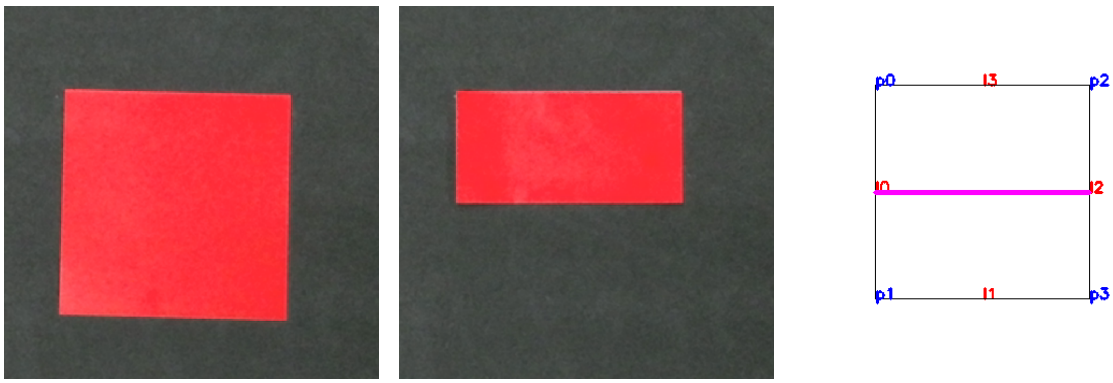



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.16: (1) の折り動作推定結果

表 3.1: (1) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 1 | 1 | 2 | |
| 1 | 2 | 1 | |
| 2 | 3 | 0 | |
| 2 | 0 | 3 | |
| 3 | 1 | 0 | |
| 3 | 0 | 1 | |
| 3 | 2 | 3 | |
| 3 | 3 | 2 | |
| 5 | 1 | 3 | 0 |
| 5 | 1 | 3 | 3 |
| 5 | 1 | 0 | 2 |
| 5 | 1 | 0 | 1 |
| 5 | 2 | 3 | 0 |
| 5 | 2 | 3 | 3 |
| 5 | 2 | 0 | 2 |
| 5 | 2 | 0 | 1 |

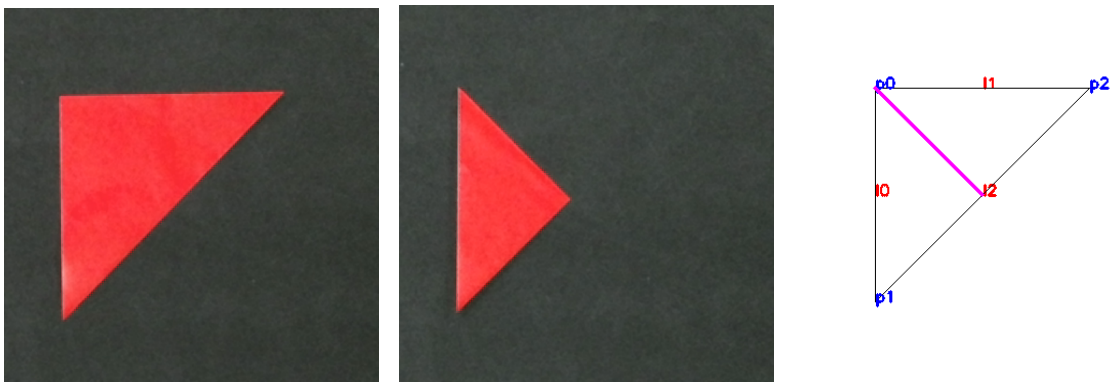



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.17: (2) の折り動作推定結果

表 3.2: (2) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 0 | |
| 2 | 0 | 1 | |
| 2 | 3 | 2 | |
| 2 | 2 | 3 | |
| 3 | 1 | 3 | |
| 3 | 3 | 1 | |




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

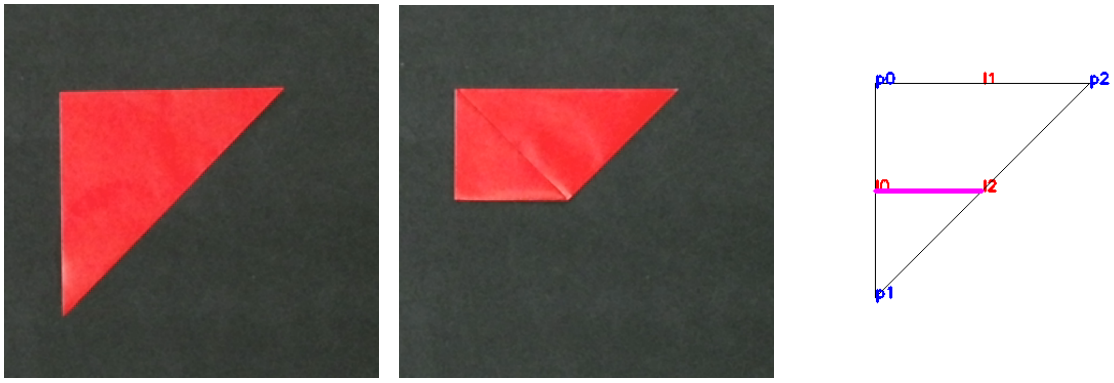
図 3.18: (3) の折り動作推定結果

表 3.3: (3) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 2 | 1 | |
| 3 | 1 | 0 | |
| 4 | 0 | 2 | |
| 5 | 0 | 2 | 0 |
| 5 | 0 | 1 | 1 |

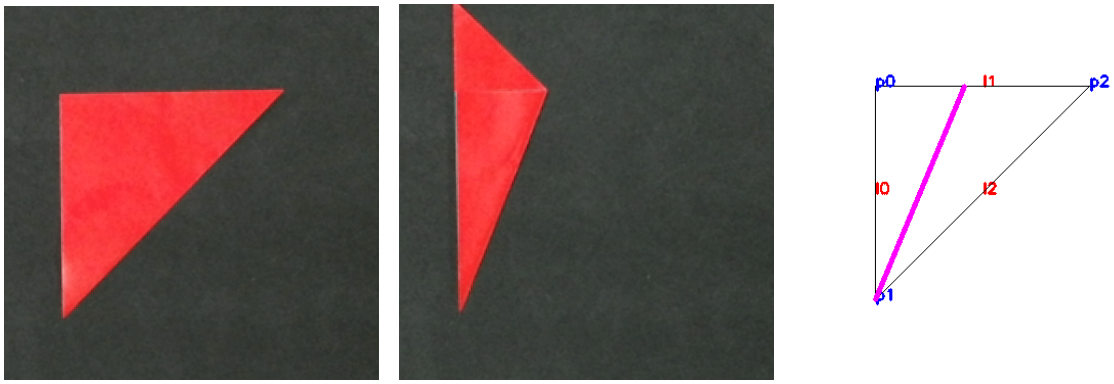
表 3.4: (4) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 0 | |
| 2 | 0 | 1 | |




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.19: (4) の折り動作推定結果




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

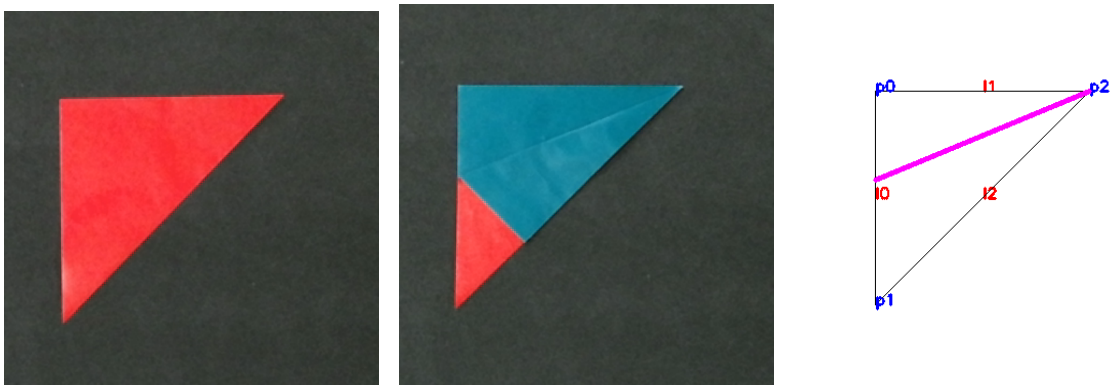
図 3.20: (5) の折り動作推定結果

表 3.5: (5) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 2 | 0 | |
| 3 | 0 | 2 | |
| 5 | 1 | 0 | 2 |

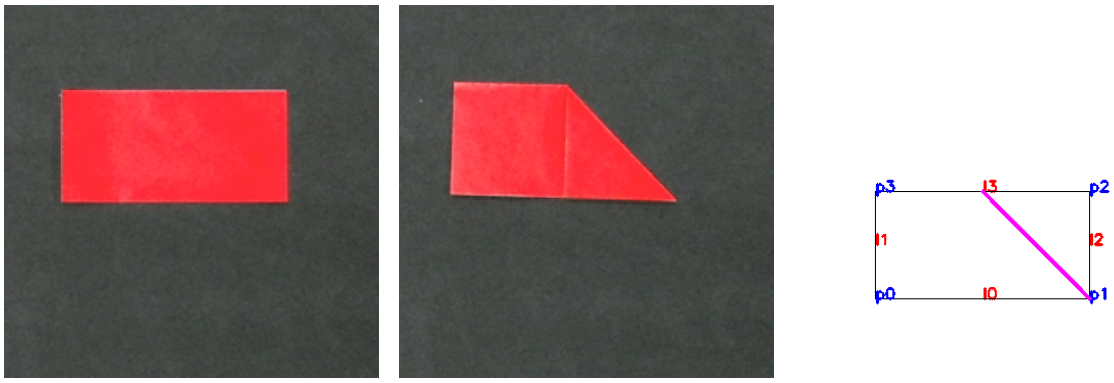
表 3.6: (6) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 3 | |




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.21: (6) の折り動作推定結果

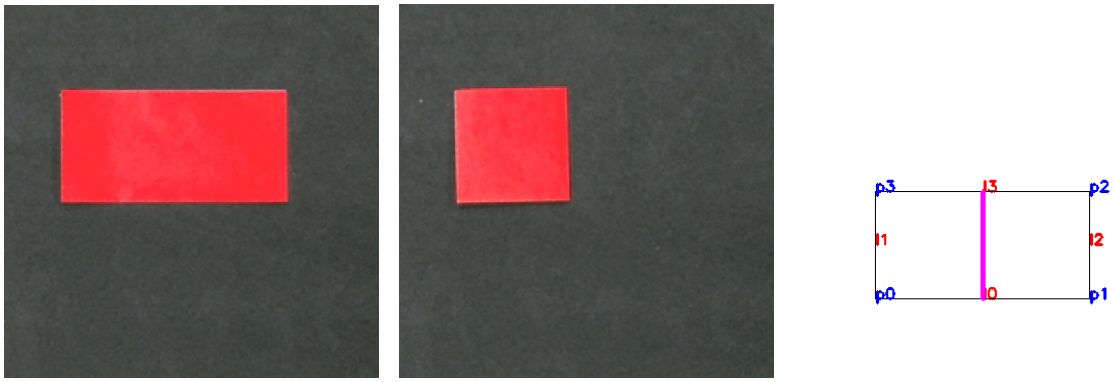



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.22: (7) の折り動作推定結果

表 3.7: (7) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 2 | 0 | |
| 3 | 0 | 2 | |
| 5 | 1 | 2 | 0 |

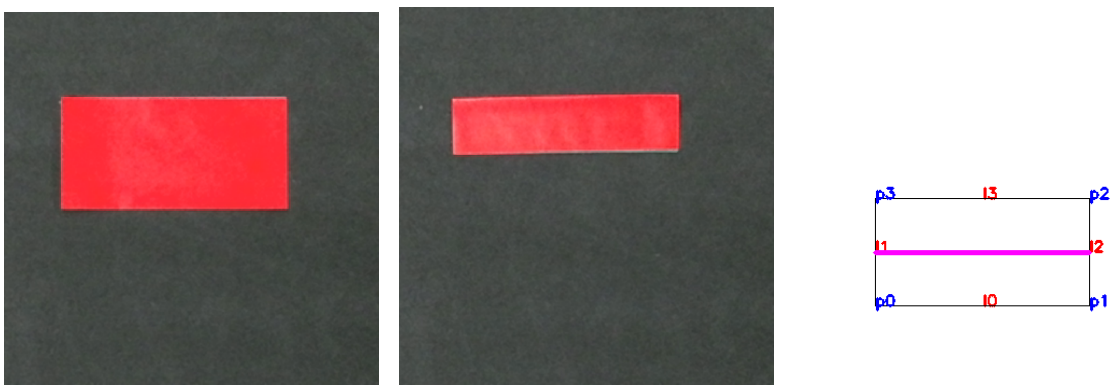



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 3.23: (8) の折り動作推定結果

表 3.8: (8) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 0 | |
| 2 | 0 | 1 | |
| 2 | 2 | 3 | |
| 2 | 3 | 2 | |
| 3 | 2 | 1 | |
| 3 | 1 | 2 | |

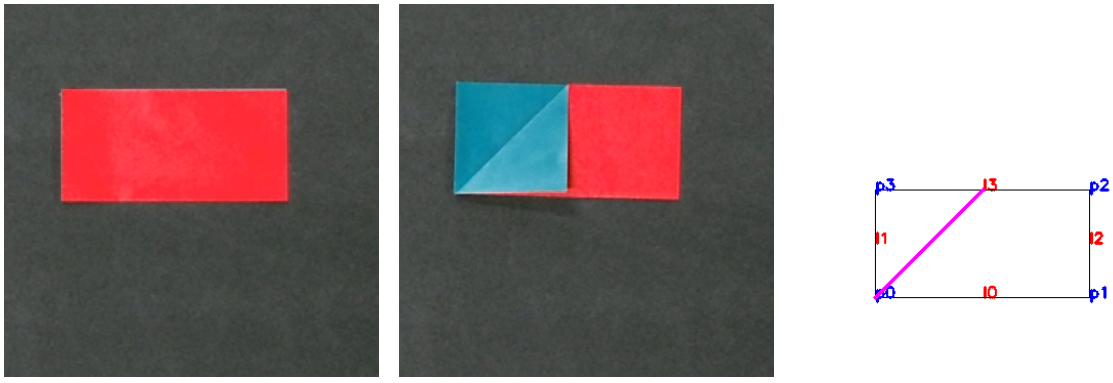



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 3.24: (9) の折り動作推定結果

表 3.9: (9) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 0 | 3 | |
| 2 | 3 | 0 | |
| 2 | 1 | 2 | |
| 2 | 2 | 1 | |
| 3 | 0 | 3 | |
| 3 | 3 | 0 | |

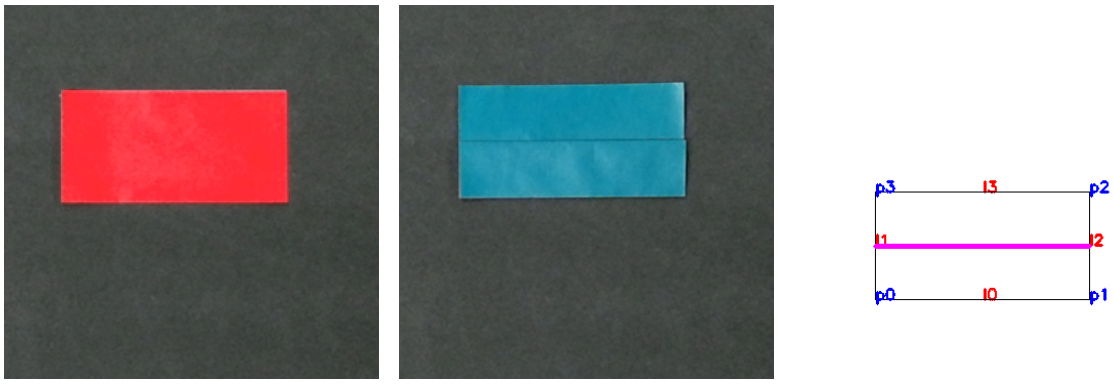



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.25: (10) の折り動作推定結果

表 3.10: (10) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 1 | 0 | |
| 3 | 0 | 1 | |
| 5 | 0 | 3 | 0 |

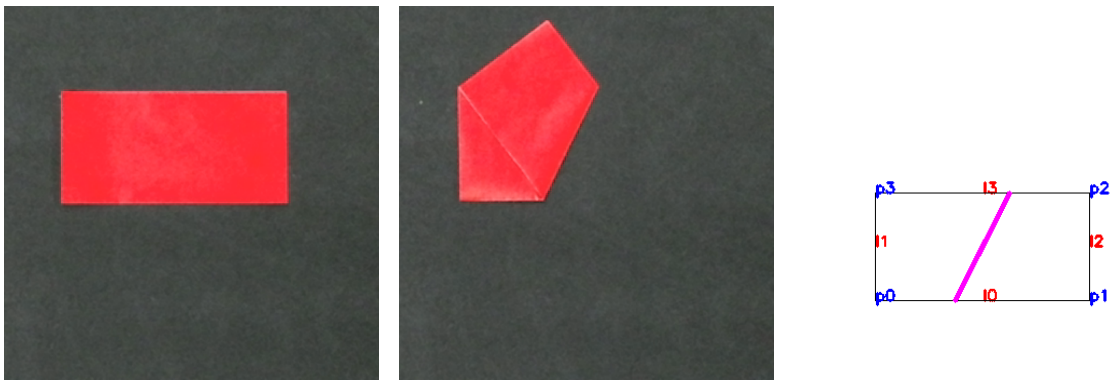



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.26: (11) の折り動作推定結果

表 3.11: (11) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 2 | |
| 2 | 2 | 1 | |
| 2 | 0 | 3 | |
| 2 | 3 | 0 | |
| 3 | 0 | 3 | |
| 3 | 3 | 0 | |

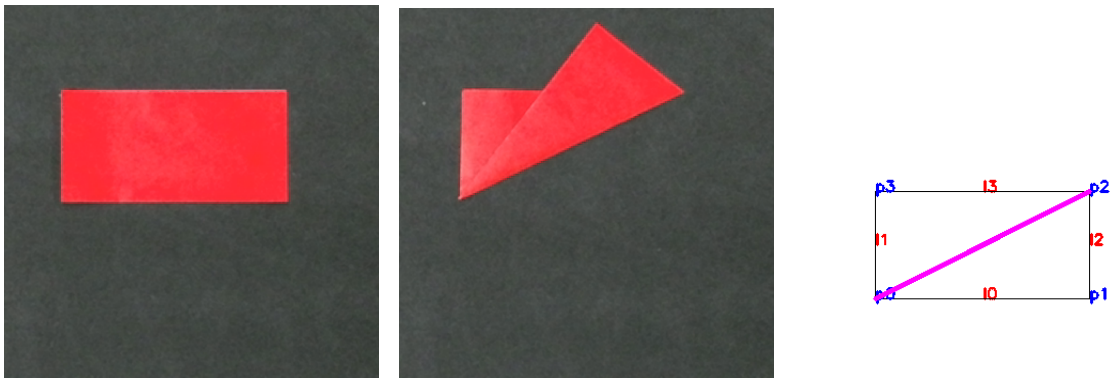



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.27: (12) の折り動作推定結果

表 3.12: (12) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 3 | |
| 2 | 3 | 1 | |

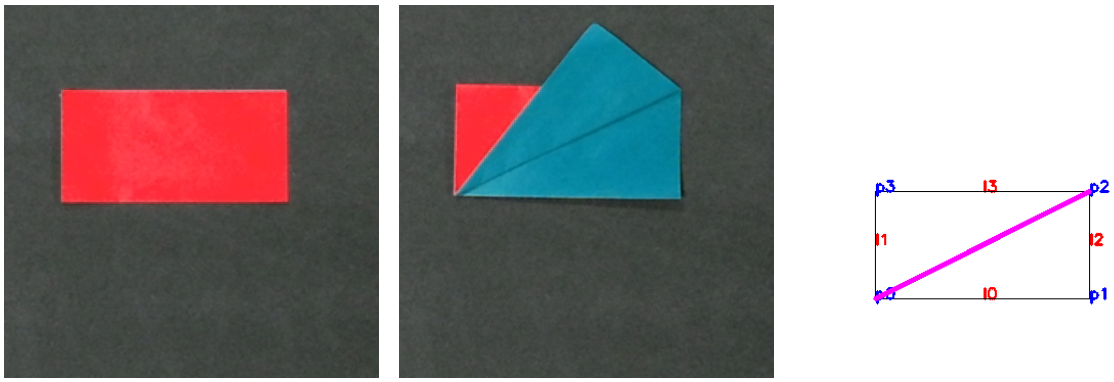



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.28: (13) の折り動作推定結果

表 3.13: (13) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 1 | 0 | 2 | |
| 1 | 2 | 0 | |

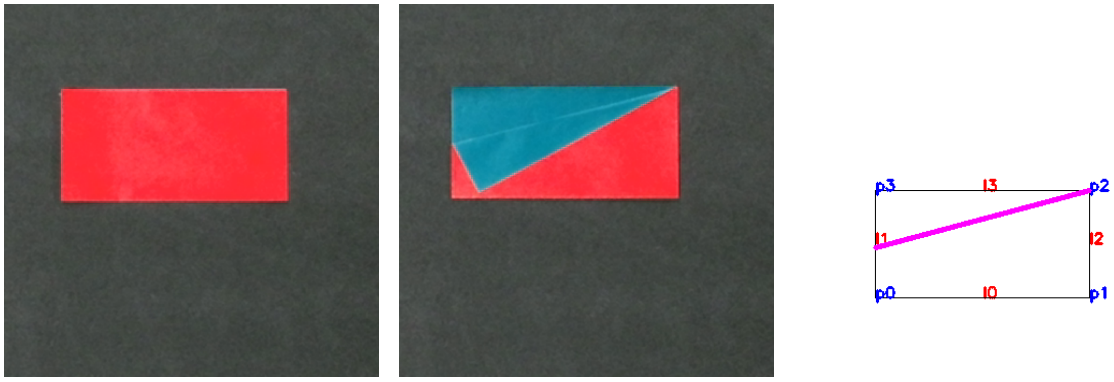



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.29: (14) の折り動作推定結果

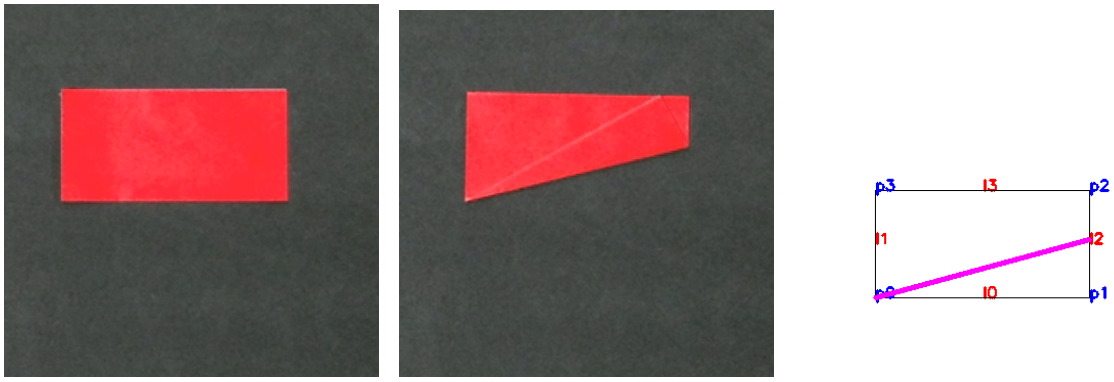
表 3.14: (14) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 1 | 0 | 2 | |
| 1 | 2 | 0 | |




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.30: (15) の折り動作推定結果




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 3.31: (16) の折り動作推定結果

表 3.15: (15) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 5 | 2 | 3 | 0 |

表 3.16: (16) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 5 | 0 | 1 | 3 |

### 3.4.2 計算時間と列挙された候補数

本章のアルゴリズムでは，ICP アルゴリズムにより，位置を推定しているため，計算時間が多く，リアルタイムな処理では行えない可能性が考えられる．そこで，どれだけの時間が処理に必要なのかを実験により確かめた．以下には計算に必要となった時間を折り作品ごとに表としてまとめ，表 3.4.2 に記す．

平均時間は 19sec であり，標準偏差は 0.275sec であった．計算時間を検討する限り，回の

表 3.17: 計算時間

| ラベル | 計算時間（sec） |
|---|---|
| 1 | 19 |
| 2 | 19 |
| 3 | 18 |
| 4 | 19 |
| 5 | 19 |
| 6 | 19 |
| 7 | 18 |
| 8 | 18 |
| 9 | 18 |
| 10 | 19 |
| 11 | 19 |
| 12 | 19 |
| 13 | 18 |
| 14 | 18 |
| 15 | 18 |
| 16 | 18 |

操作推定のために平均 20sec 近くかかっている．これだけの時間がかかる要因は ICP アルゴリズムが原因である．しかし，本研究の目的は，あくまで折り紙の操作を認識するための物であり，リアルタイム性は求めるものではないため，特に問題とはしないものとする．今後，リアルタイムな処理が必要とされる場合には ICP アルゴリズムとは別の手法を用いて折り紙の位置を検出する手法の検討が必要であると考える．

### 3.4.3 まとめ

上記により，本アルゴリズムでは，折り紙の折り作業を推定可能であることを示した．しかし，本アルゴリズムにおいて 3 回折り以降では，第 3.3.1 節で記述する折り方のパターンで示しきれない折り方が複数存在している．また，折り紙の位置が少しでもずれた場合，折り線の検出は困難となる．次章では，それら問題に対応するアルゴリズムを検討する．

# 第4章　折り線候補群を用いた認識手法

第3章では，折り紙の動作前後で折り紙のある頂点が移動および回転しないものとしていた．そのため，折り紙が移動することに画像認識を用いた折り線検出が大きな影響を受けていた．また，折り線検出のアルゴリズムの構造上，3回以降の折り方では検出が困難になることが予想されていた．本節では，そういった問題を解決するための手法を提案する．以下では，まず折り紙の画像を撮影するシステムについて記述する．その後，アルゴリズムの概要と詳細を記述する．最後に検討したアルゴリズムに対し実験を行う．

## 4.1　折り紙画像の撮影システム

本章の内容は第3.1節と同じ内容とする．

## 4.2　アルゴリズムの概要

3章で折り紙が移動できない主な原因として折り線検出の手法が問題であった．しかし，折り紙に自由な移動を許した場合，画像認識による折り線検出手法では折り線の検出が困難であり，また折り動作前と後では大きく形状が変わる場合が多い．折り動作前後の折り紙の状態の位置関係を調べる必要がある．その場合，考えられる手法としてICPアルゴリズムのような位置合わせアルゴリズムが存在する．こういったアルゴリズムは折り動作前後で一部でも同じ部分があればそこに位置合わせすることが可能であるが，初期値依存性等により必ずしも折り動作前後で変化しない部分に移動できるわけではない．また，共通部分となる場所は折り紙によりオクルードされる可能性があるほか，反転している可能性も存在する（通常のICPでは反転に対応していない）．これらの要因から，折り動作前の画像上での位置と折

り動作後の画像上での位置の対応関係をとることは困難であると考える．そこで，本研究では，折り動作後の状態を推定し，推定した結果と折り動作後の折り紙の画像との位置関係を推定する手法を提案する．

まず，折り動作後の状態を推定する手法を提案する．検討にあたり，折り紙公理の特性に着目し，折り動作後の状態を推定することとした．折り紙公理では点や線の情報があればそこから折り線を計算でき，また折り線を作成する際に使用した点や線の情報を保持することができる．つまり，折り紙公理では点と線の集合から折り紙公理で記述可能な全ての作業を列挙できるといえる．さらに，この場合の作業とは生成する折り線の位置と折り線を作るために用いられた公理の内容となる．

そこで，先行研究である [20] で用いた手法を利用する．先行研究では，折り紙の折り動作前の状態とそこに適用される折り線の位置が入力として存在すれば，そこから折り動作後の折り紙と同じ形を含む全ての折り紙公理により記述可能な作業結果の形状が列挙可能である．

その特性を用いて，本研究では，折り動作前の折り紙の状態において折り紙公理で記述可能な全ての折り線を列挙し，それら折り線で作成可能な全ての折り結果を列挙する．そして，列挙された状態の中から折り作業の行われた結果の形状に近いものを選び出すことでこの問題に対応することとした．

以下にそのアルゴリズムのフローを示し，図 4.1 に図示する．

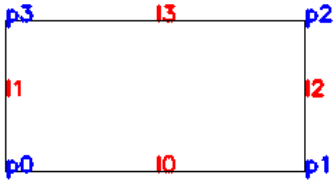
Step1 折り動作前の折り紙の辺や頂点の組み合わせをすべて列挙し，それらの組み合わせに対し，折り紙公理を適用し，折り線候補を列挙する．

Step2 列挙された折り線で可能な折り動作後の状態を第 3.3.2 節のアルゴリズムを用いてすべて列挙

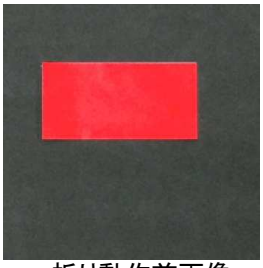
Step3 列挙された状態の中から，第 3.3.2 節のアルゴリズムを用いて 1 番確からしいものを選ぶ．

以下にこれらの概要を記述する．

入力

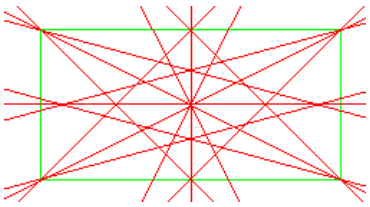



折り動作前状態

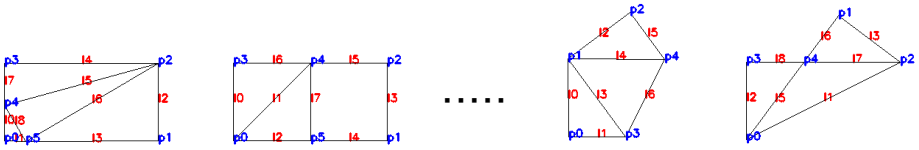


折り動作前画像

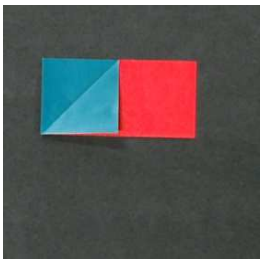
Step1:可能な折り線を全て列挙



Step2:折り動作後の状態を列挙



Step3:折り動作後の形状に近いものを選択



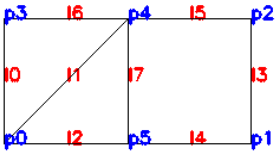



図 4.1: 折り線候補群を用いた認識手法

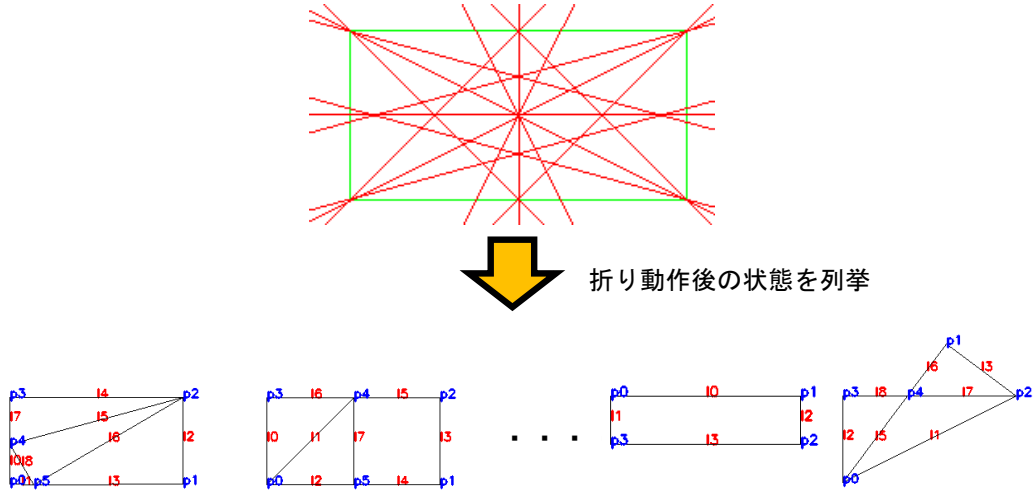



図 4.2: 状態列挙アルゴリズム

### 4.2.1 折り線列挙アルゴリズム

折り紙公理では公理ごとに必要となる点や線の情報から折り線を生成可能である．本節ではその特性を用いて折り紙公理で記述可能な全ての折り線を折り紙公理ごとに列挙するアルゴリズムを検討する．各公理ごとに作成された折り線の情報はそれぞれ折り線上の任意の 2 点の情報を格納する動的配列 orisen_info と，その折り線に必要な折り紙公理の情報を Axiom_Info の axiom_infos と同じように保存する二重動的配列に保存される．また，折り紙公理では折り線は直線として扱うため，実際は 2 つの点として記述するが以降の処理では直線であるとして処理を行う．

### 4.2.2 状態列挙アルゴリズム

本節では，前節で列挙された折り線から生成可能な折り紙の状態を列挙することを目的とする．本節の内容は基本的に第 3.3.2 節と同じ内容である．差異の部分としては，本章のアルゴリズムでは折り線の数は一つではなく，複数である点である．以下に検討した手法のフローを記述し，図 4.3 に図示する．

### 4.2.3　状態推定アルゴリズム

本節では，前節で列挙された折り紙の状態から折り動作後の折り紙の形状に近い状態を選び出すことを目的とする．本節の内容はいくつか第 3.3.2 節と同じ内容が存在する．差異の部分としては，本章のアルゴリズムでは折り紙の移動を考慮している．そのため，折り紙の位置を推定するアルゴリズムとして，ICP マッチングを用いて折り動作後の折り紙の位置の対応関係の取得を行っている部分である．以下に検討した手法のフローを記述し，図 4.3 に図示する．

1. 折り紙状態の絞り込み

   列挙された折り紙の状態に対し，面積を比較することで，ありえないと思われるものを省く．

2. 折り紙状態の推定

   省かれた折り紙の状態に対し，折り動作後の折り紙の形状と比較することで，折り紙の状態を推定する．

## 4.3　アルゴリズムの詳細

### 4.3.1　折り線列挙アルゴリズム

前述のとおり，折り線を列挙するアルゴリズムを公理ごとに記述する．

#### 公理 1

公理 1 とは 2 点 $p_1, p_2$ から折り線を定義する．そのため，State の全ての頂点の組み合わせに対して折り線を列挙する．また，この公理は逆順でも成り立つため，両方の順序の物を生成する．以下にその処理を記述する．

1. Axiom_Info 構造体の変数を定義し，折り線の頂点を $p_1$，$p_2$ とし，公理の情報として $p_1$，$p_2$ の情報を保存する．

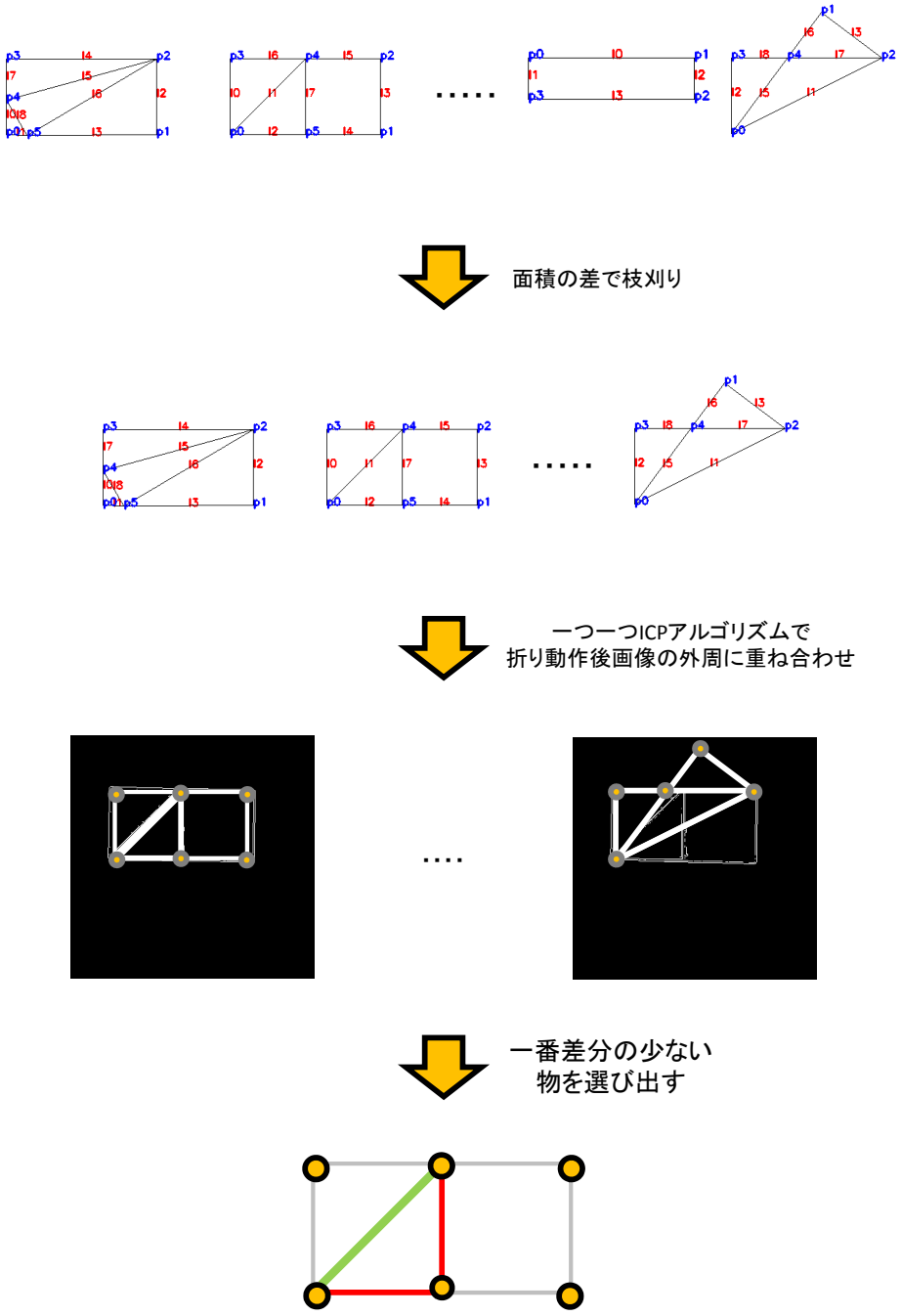



図 4.3: 状態推定アルゴリズム

**公理 2**

公理 2 により作られる折り線は，与えられた 2 点 $p_1, p_2$ の垂直ニ等分線により生成される．そのため，State の全ての頂点の組み合わせに対して折り線を列挙する．この時，頂点の各組合せに対し，以下のような処理を行うことで折り線を出力する．

1. $p_1$，$p_2$ の重心点 $p_{center}$ を求める．
2. $p_{center}$ を中心に，$p_1$ を 90 度回転した点 $p_{turned}$ を求める．
3. Axiom_Info 構造体の変数を定義し，折り線の頂点を $p_{center}$ と $p_{turned}$ とし，公理の情報として $p_1$，$p_2$ の情報を保存する．

**公理 3**

公理 3 により作られる折り線は，2 線 $l_1, l_2$ の 2 等分線により生成される．そのため，State の全ての辺の組み合わせに対して折り線を列挙する．また，ニ等分線の生成には 2 線のベクトルを生成し，その和のベクトルとそれに直行するベクトルの 2 つが可能なため，それら両方を折り線として出力する．

この時，辺の各組合せに対し，以下のような処理を行うことで折り線を出力する．

1. 2 線 $l_1, l_2$ のベクトル $Vl_1, Vl_2$ を求める．
2. ベクトル $Vl_1, Vl_2$ の和 $V_{sum}$ を求める．
3. 2 線 $l_1, l_2$ の交点 $Cross_V$ を求める．
4. $Cross_V$ を開始点として $V_{sum}$ 分進んだ点を計算する $P_{candidate1}$．
5. $Cross_V$ を中心として $P_{candidate1}$ を 90 度回転した点 $P_{candidate2}$ を求める．
6. $p_{center}$ を中心に，$p_1$ を 90 度回転した点 $p_{turned}$ を求める．
7. 折り線の頂点に $p_{center}$ と $p_{turned}$ をとする．

8. Axiom_Info 構造体の変数を定義し，折り線の頂点を $\mathrm{Cross_V}$ と $\mathrm{P_{candidate1}}$ とし，公理の情報として $\mathrm{l_1}$，$\mathrm{l_2}$ の情報を保存する．

9. Axiom_Info 構造体の変数を定義し，折り線の頂点を $\mathrm{Cross_V}$ と $\mathrm{P_{candidate2}}$ とし，公理の情報として $\mathrm{l_1}$，$\mathrm{l_2}$ の情報を保存する．

**公理 4**

公理 4 は与えられた 1 点 p と 1 本の，l に対し，点を通るように線を自分自身の上に重ねて，与えられた点を通るように与えられた線の垂直ニ等分線で折る．折り線は，p と p から l に下した垂線との交点 $\mathrm{P_{vertical\_cross}}$ で得られる．

1. p から l に下した垂線との交点 $\mathrm{p_{vertical\_cross}}$ を計算する．

2. Axiom_Info 構造体の変数を定義し，折り線の頂点を p，$\mathrm{P_{vertical\_cross}}$ とし，公理の情報として p，l の情報を保存する．

**公理 5**

公理 5 は与えられた 2 点 $\mathrm{p_1, p_2}$ と 1 本の線 l に対し，一方の点 $\mathrm{p_2}$ が一方の線 l に乗るように他方の点 $\mathrm{p_1}$ を通る線で折る．折り線の計算のためには $\mathrm{p_2}$ を中心とし，$\mathrm{p_1}$ と $\mathrm{p_2}$ の間の距離の半径を持つ円と，l の交点を求める．そのため，2 つの候補が出てくるが，その両方を出力する．以下のそれらのアルゴリズムを示す．

1. $\mathrm{p_1}$ と $\mathrm{p_2}$ の間の距離を計算し，r とする．

2. $\mathrm{p_2}$ を中心とし r の半径を持つ円と l の交点 $\mathrm{p_{cross1}}$，$\mathrm{p_{cross2}}$ を求める．

3. 交点が 1 つであれば，その交点と $\mathrm{p_1}$ を結果として出力する．Axiom_Info 構造体の変数を定義し，折り線の頂点に交点と，$\mathrm{p_1}$ を，公理の情報として $\mathrm{p_1, p_2}$ と l の情報を保存して終了する．

4. 交点が 2 つの時次の処理を行う．

   (a) Axiom_Info 構造体の変数を定義し，折り線の頂点を $p_{cross1}$ とし，$p_1$ を，公理の情報として $p_1, p_2$ と l の情報を保存する．

   (b) Axiom_Info 構造体の変数を定義し，折り線の頂点を $p_{cross2}$ とし，$p_1$ を，公理の情報として $p_1, p_2$ と l の情報を保存する．

5. $p_1$ を通り $p_{cross1}$ または $p_{cross2}$ を通る折り線の内，3.3.1 で検出された折り線に近い交点 $p_{nearcross}$ を求める

6. 折り線の頂点を $p_1$ と $p_{nearcross}$ とする．

### 折り線の統合

4.3.1 で得られる折り線は生成するための公理は違うが同じ折り線が複数存在する．また，実際には特に公理 5 などで折り紙上に折り線が被ることがない折り線も生成される．そのため，折り紙上に被らない折り線を検出し，省く必要がある．本節では操作不能な折り線を省きつつ，同じ折り線を生成可能な公理を 1 つにまとめる手法について記述する．まとめた公理は，Folding_Operation 構造体の axiom_infos に保存される．以下にその手法のフローを示す．

1. 計算された全ての折り線（All_Axiom_Line）に対し，以下の手順を行うことで折り紙上にない折り線の排除を行う．

   (a) ある折り線のみを描画した二値画像 O を作成する．

   (b) 画像 A に対し，折り紙の部分のみ白くなるよう二値化した画像 $A_b$ を作成

   (c) $A_b \bigcap O$ となる部分の 2 値画像 $O_o$ を作成する．

   (d) $O_o$ の白い部分の面積を求め，それが 0 ならばその折り線は All_Axiom_Line から排除する

2. 結果を格納するための構造体として Folding_Operation 構造体の動的配列を Axiom_operations とする．

3. All_Axiom_Line の全ての折り線に対し，以下の処理を行う．

   (a) All_Axiom_Line から順に折り線を取り出す（One_Axiom_Line）

   (b) 既に Axiom_operations に追加されている場合は 1 に戻る．

   (c) Folding_Operation 構造体の One_Operation を作成し One_Axiom_Line を axiom_infos に追加する．

   (d) 他の全ての折り線に対し，以下の処理を行う．

      i. One_Axiom_Line からの角度を求める（line_angle）

      ii. One_Axiom_Line に対し，原点から垂線を下ろし，その垂線の距離を求める（Perpendicular_range1）

      iii. 他の全ての折り線に対し，原点から垂線を下ろし，その垂線の距離を求める（Perpendicular_range2）

      iv. Perpendicular_range1 と Perpendicular_range2 の値が近く，line_angle の値が小さい物は，One_Operation に追加する．

   (e) One_Operation を Axiom_operations に追加する

### 4.3.2 状態列挙アルゴリズム

前述した状態列挙の流れの詳細を以下に記述する．処理の詳細において第 3.3.2 節のアルゴリズムの部分は割愛する．

1. 第 4.3.1 節のアルゴリズムを用いて列挙した折り線一つ一つに対して以下の処理を行う

   (a) 第 3.3.2 節のアルゴリズムを用いて折り動作前の状態に対して可能な全ての折り紙の状態を列挙する．

### 4.3.3 状態推定アルゴリズム

前述した状態推定の流れの詳細を以下に記述する．

1. 折り紙状態の枝刈り

   列挙された折り紙の状態候補群に対し，面積を比較することで，ありえないと思われるものを省く．以下にその処理を記述する．

   (a) 全ての折り紙の状態候補群に対し，以下の処理を行う．

      i. 入力された折り動作後の折り紙の 2 値画像 $B_b$ に対し，白い部分の面積（area）を求める．

      ii. ある折り紙の状態候補 $State_{tmp}$ に対し，辺をつなぐように線を描画した 2 値画像 $SV_{tmp}$ を作成する．

      iii. floodFill アルゴリズムにより，内部を白で埋めた画像を作成し，白い部分の画像の面積を求め，area と大きく値が異なる場合は排除する．

2. ICP アルゴリズムにより折り紙位置の探索

   枝刈りされたすべての折り紙の状態に対し，ICP アルゴリズムにより位置を探索した．

3. 折り動作後の折り紙の状態に近いものの選出

   位置の特定された折り紙の状態 1 つ 1 つに対して推定し，第 3.3.2 節のアルゴリズムにより誤差値を計算し，1 番誤差の少ないものを正しい折り動作の結果として出力した．

4. 視覚的な交点の検出

   (a) 選び出された状態 $State_{new}$ に対し，$State_{new}$ 上の線全ての交点を計算し，$State_{new}$ の点の近くにない交点を計算する．

   (b) $State_{new}$ 上で $V_{dev}$ の重なる線を $V_{dev}$ で分断し，2 つの線とする．

## 4.4 実験

実験では，本節のアルゴリズムについて 2 回折りまでの折り紙に対し，1 回ずつ実験を行い，本節のアルゴリズムが正常に記述できるかを示した．個々での実験結果であるが，折り線の位置及び行われた動作については第 3.4.1 節の結果と全く同じになったため，割愛する．以下には 2 回折りにおける計算時間と，3 回折りにおける実験結果について記述する．

### 4.4.1 計算時間

本章のアルゴリズムでは，ICP アルゴリズムを列挙された全ての候補に対して行うため，第 3.4.2 章の計算時間より遥かに多くの計算時間が必要なことが見込まれる．また，必要な計算時間は列挙される折り紙の状態候補の数により変動することが予想される．そのため，表 4.1 は各折り作品ごとの計算時間と，候補数を記載する．上記アルゴリズムは平均値 68sec

表 4.1: 各ラベルごとの計算時間及び列挙された折り紙の状態候補数

| ラベル | 計算時間 (sec) | 列挙された候補数 |
|---|---|---|
| 1 | 83 | 16 |
| 2 | 87 | 16 |
| 3 | 21 | 4 |
| 4 | 32 | 8 |
| 5 | 46 | 12 |
| 6 | 25 | 6 |
| 7 | 171 | 48 |
| 8 | 33 | 8 |
| 9 | 33 | 8 |
| 10 | 47 | 12 |
| 11 | 47 | 12 |
| 12 | 116 | 32 |
| 13 | 61 | 20 |
| 14 | 75 | 20 |
| 15 | 47 | 12 |
| 16 | 171 | 48 |

に対し，標準偏差 45sec であった．つまり，非常に各ラベルごとに計算時間が異なる結果となった．また，列挙された候補数 1 つ当たりでは，平均値が 4sec であり標準誤差 0.653sec であった．これらより本アルゴリズムでは列挙される候補数に従い，計算量が増大していくことが示された．候補数の増大は前述の通り，折り紙の点や線が増えるに従い階乗的に増加するため，計算量の検討は必須であると考えられる．

### 4.4.2 3 回折りにおける実験結果

本節では，第 3 章で記述された．20 個の 3 回目の谷折操作で記述可能な折り紙作品に対し，本アルゴリズムによる実験を行い，システムの評価を行う．実験結果は第 3.4.1 節と同じように検出された折り線の位置と行われた操作で記述する．実験結果として，20 個中，17 個において正しく折り作業を認識できたが (17)，(21), (25）においては正しく折り作業の認識をすることはできなかった．認識できたもののうち，特に（22）, (30）, (33）は第 3.3.1 節での折り方のパターンで表せないものである．この結果により，本アルゴリズムは第 3.3.1 節の問題を解消できている可能性があることが示される．

今回失敗した 3 つの折り作品に対し，これら特徴ごとにアルゴリズムの問題点が 2 つ上げられる．それらについて以下で考察する．

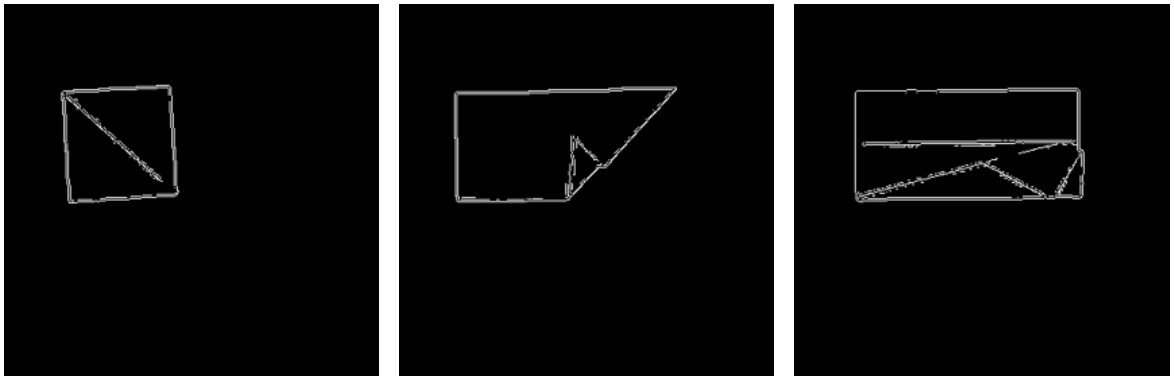
1. 点や線の数の影響

   今回失敗した 3 つのラベルであるが，これらにある特徴として折り動作前の状態の点の数が多いものほど失敗しているということがあげられる．以下に折り動作前の状態である, (4) , (6) , (13) , (15）の点と線の数について表 4.2 にまとめる．

   本アルゴリズムの問題点は，点と線の数が増える場合に折り紙の状態候補が階乗的に増加することである．実際に，今回（4）, (6）, (13）, (15）から派生する折りで，失敗の発生していたのは点や線の比較的多い（13）, (15）であった．特に，一番点と線の数の多い (7) の折り方では，最大約 2000 個近い候補が列挙されており，今後 3 回折り以上を検討する際には，候補の枝刈り手法を検討する必要が考えられる．そういった

表 4.2: 2 回折りの時点での点と線の数

| ラベル | 点の数 | 線の数 | 点と線の合計 |
|---|---|---|---|
| 4 | 5 | 4 | 9 |
| 6 | 7 | 5 | 12 |
| 15 | 9 | 6 | 15 |
| 13 | 6 | 5 | 11 |



(a) (17)　　(b) (21)　　(c) (25)

図 4.4: (17)，(21)，(25) のエッジ画像

ことから本アルゴリズムは，認識する動作前の折り紙の点と線の増加に対し，精度が低下するといえる．

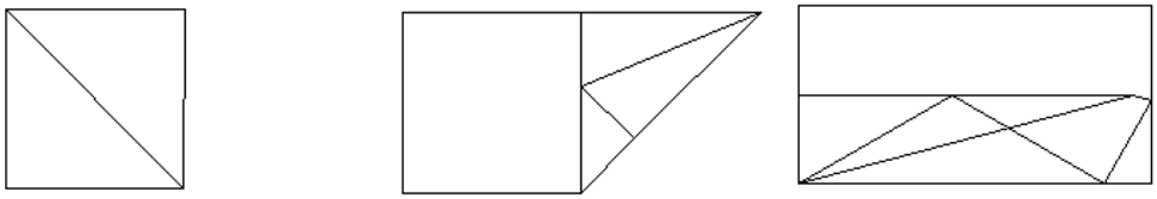
2. 特徴点で見えないエッジが発生する場合

本アルゴリズムでは，すべてのエッジが見えていればマッチングできるが，見えていない場合マッチングできない可能性が存在する．図 4.5 に（17）,（21）,（25）のエッジ抽出結果を示す．

見えない要因として次の二つが考えられる．

(a) 画像処理でエッジが見えていない

本アルゴリズムでは画像処理でエッジを抽出している．その際に，全てのエッジを抽出できていない場合が存在する．実際に,（21）のラベルではエッジが一部抽出されてなく，それが原因でミスマッチが発生したと考えられる．



(a) (17)　　(b) (21)　　(c) (25)

図 4.5: (17)，(21)，(25) の理想的な状態画像

(a) (17)　　(b) (21)

図 4.6: (17)，(21) の状態を可視化した画像

(b) オクルードされたエッジが存在する

推定される折り紙の状態では図 4.5(c) のようにオクルードされたエッジも全て記述される．しかし，撮影される折り紙の画像はオクルードされた部分が写っておらず，またそれを予想するアルゴリズムを用いてエッジを検出しているわけではない．そのため，マッチングの際にはその部分の差によりミスマッチが発生している．

3. 状態の絞り込みアルゴリズムの限界

今回失敗したラベルの内，(17)，(21) においては，折り線の位置が正解に近い位置にいるものの，間違えた位置に存在している．図 4.6 に推定された (17)，(21) の状態を記載する．

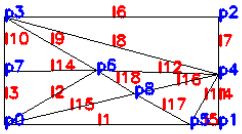
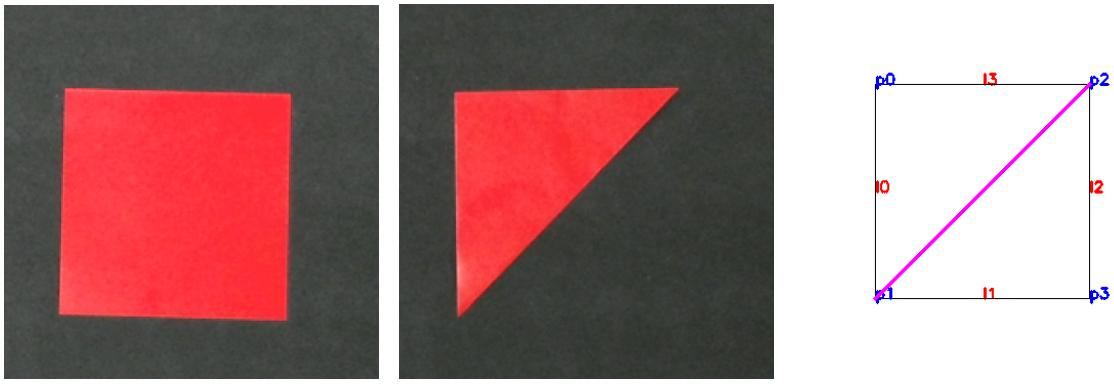



図 4.7: （25）の状態を可視化した画像

これらを見るにあたり，近い形ではあるが実際の折り紙の形とは違う形が推定されている．これらから，ICP アルゴリズムを用いた状態候補の絞り込みの際に，外輪郭の形状が似通ったものである場合に失敗が起こりやすいことが考えられる．

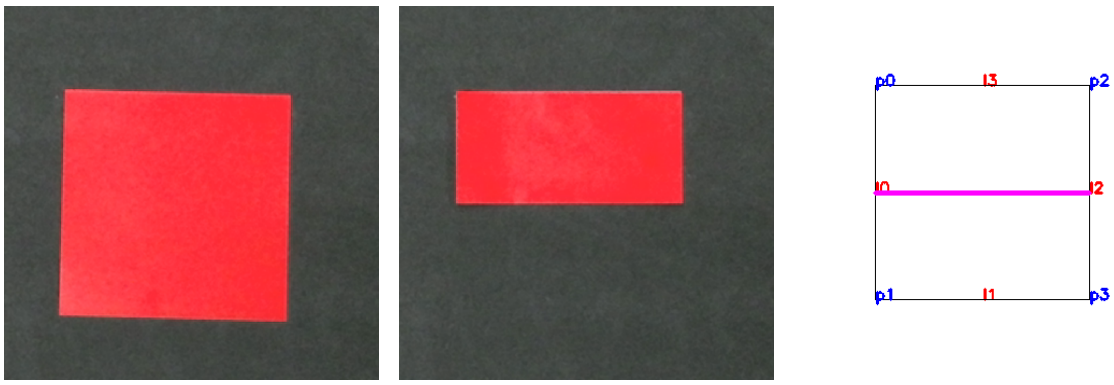
また,（25）においては，折り動作後の折り紙の画像に含まれる部分が存在しているという特徴がある．図 4.7 に推定された（25）の状態を記載する これらを実際の折り紙の画像と比べると（25）の含まれる部分が存在しており，こういったことから，ICP アルゴリズムで位置が会い，また 誤差値の推定において小さい誤差となってしまったことがあげられる．今後，誤差値の出力方法においても改善の余地が見込まれることが示されたといえるだろう．

これらの結果より，今後より複雑な折り作業の認識を行うためには，状態の絞り込みアルゴリズムの中でも特に候補の枝刈り手法や，誤差値の推定方法の改良が必要であると考えられる




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.8: (1) の折り動作推定結果




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.9: (2) の折り動作推定結果

表 4.3: (1) の操作推定結果

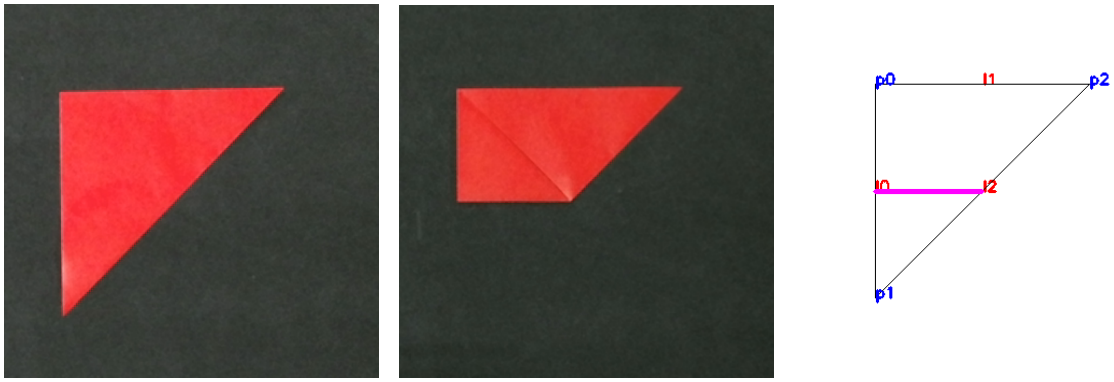
| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 1 | 1 | 2 | |
| 1 | 2 | 1 | |
| 2 | 3 | 0 | |
| 2 | 0 | 3 | |
| 3 | 1 | 0 | |
| 3 | 0 | 1 | |
| 3 | 2 | 3 | |
| 3 | 3 | 2 | |
| 5 | 1 | 3 | 0 |
| 5 | 1 | 3 | 3 |
| 5 | 1 | 0 | 2 |
| 5 | 1 | 0 | 1 |
| 5 | 2 | 3 | 0 |
| 5 | 2 | 3 | 3 |
| 5 | 2 | 0 | 2 |
| 5 | 2 | 0 | 1 |

表 4.4: (2) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 0 | |
| 2 | 0 | 1 | |
| 2 | 3 | 2 | |
| 2 | 2 | 3 | |
| 3 | 1 | 3 | |
| 3 | 3 | 1 | |

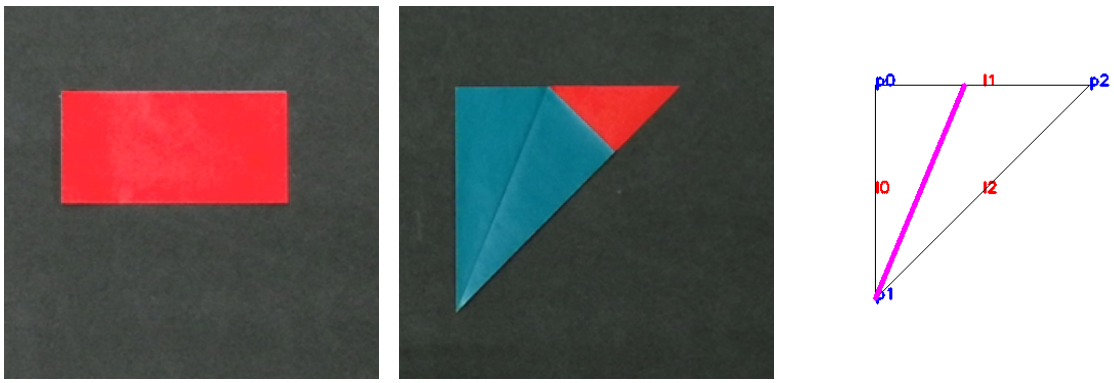
表 4.5: (4) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 0 | |
| 2 | 0 | 1 | |




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.10: (4) の折り動作推定結果

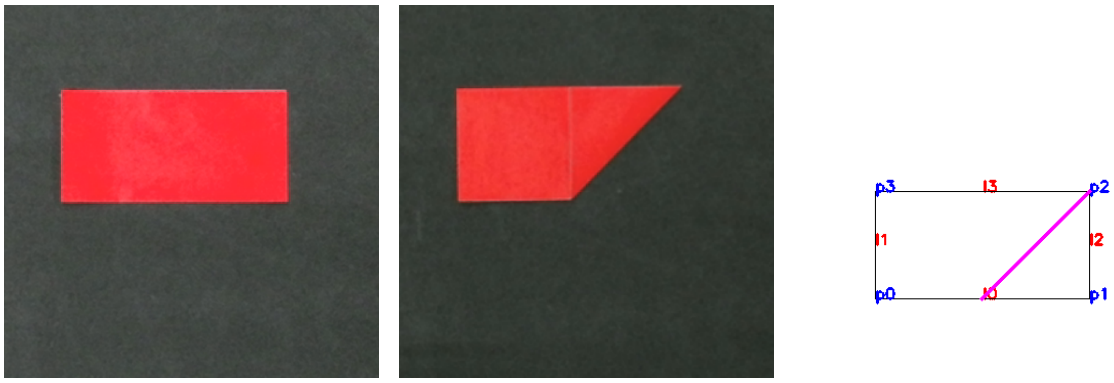



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.11: (6) の折り動作推定結果

表 4.6: (6) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 2 | 0 | |
| 5 | 1 | 0 | 2 |
| 3 | 0 | 2 | |

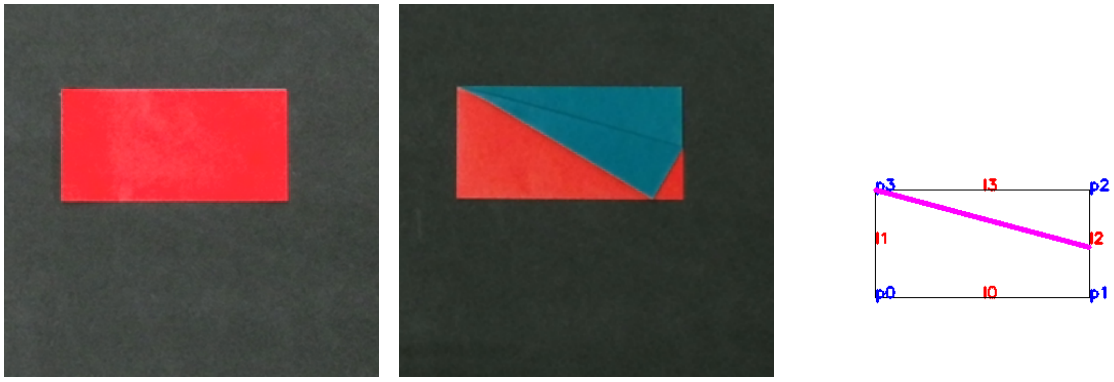



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.12: (13) の折り動作推定結果

表 4.7: (13) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 3 | 2 | |
| 3 | 2 | 3 | |
| 5 | 2 | 1 | 3 |

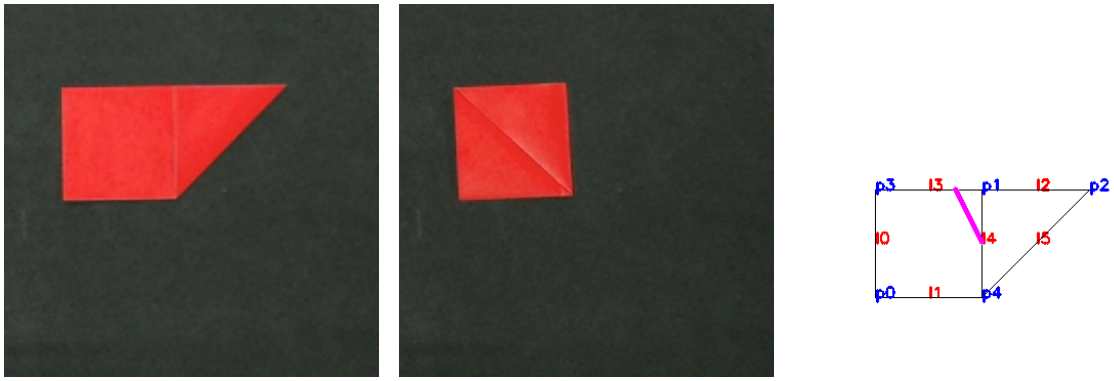



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.13: (15) の折り動作推定結果

表 4.8: (15) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 5 | 3 | 2 | 0 |

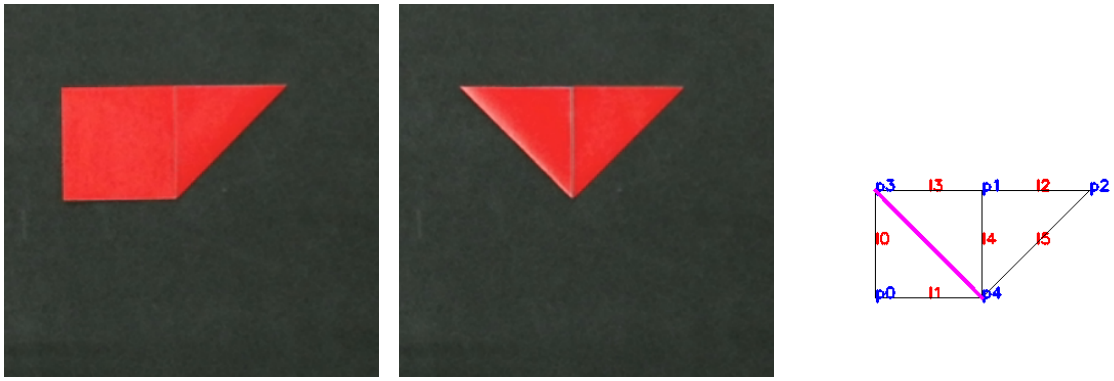



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.14: (17) の折り動作推定結果

表 4.9: (17) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 2 | 0 | |
| 2 | 0 | 2 | |

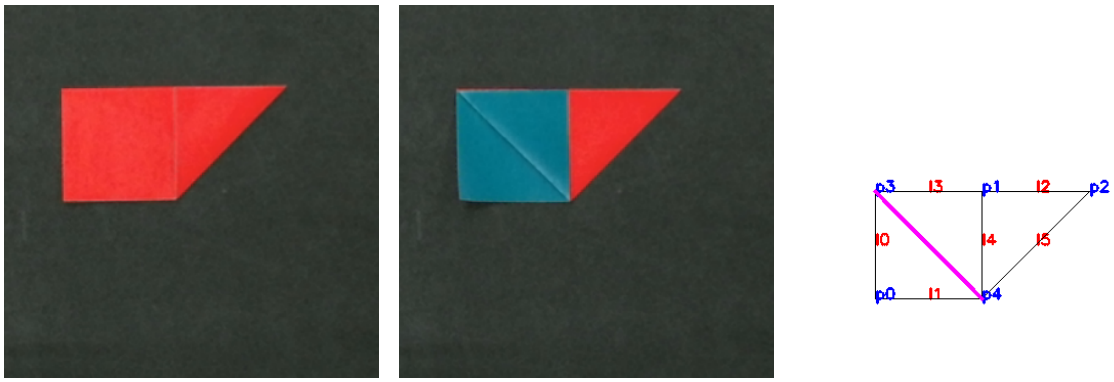



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.15: (18) の折り動作推定結果

表 4.10: (18) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 1 | 4 | 3 | |
| 3 | 0 | 2 | |
| 3 | 0 | 3 | |
| 3 | 2 | 0 | |
| 3 | 3 | 0 | |
| 4 | 3 | 5 | |
| 5 | 3 | 0 | 2 |
| 5 | 3 | 0 | 3 |
| 5 | 3 | 1 | 0 |
| 1 | 3 | 4 | |

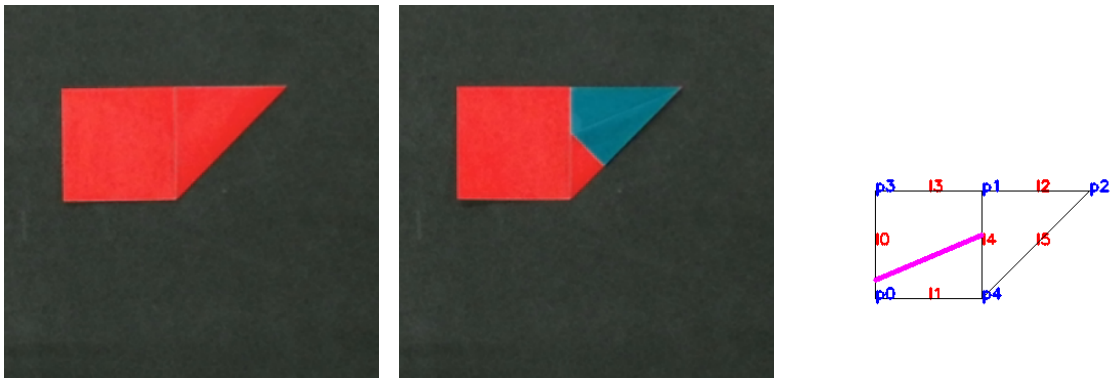



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.16: (19) の折り動作推定結果

表 4.11: (19) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 1 | 4 | 3 | |
| 3 | 0 | 2 | |
| 3 | 0 | 3 | |
| 3 | 2 | 0 | |
| 3 | 3 | 0 | |
| 4 | 3 | 5 | |
| 5 | 3 | 0 | 2 |
| 5 | 3 | 0 | 3 |
| 5 | 3 | 1 | 0 |
| 1 | 3 | 4 | |

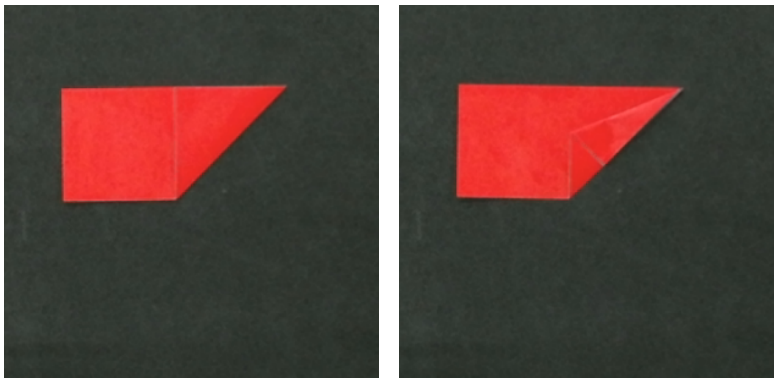



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.17: (20) の折り動作推定結果

表 4.12: (20) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 3 | 5 | |
| 3 | 5 | 2 | |
| 3 | 5 | 3 | |
| 5 | 2 | 1 | 5 |
| 5 | 2 | 4 | 3 |
| 3 | 2 | 5 | |

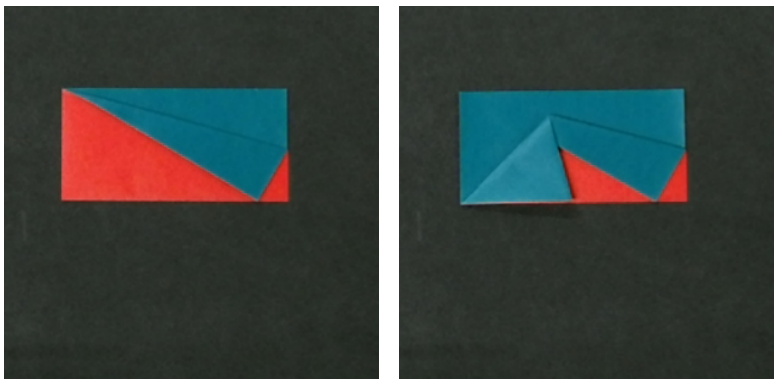



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.18: (21) の折り動作推定結果

表 4.13: (21) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 5 | 0 | 1 | 5 |

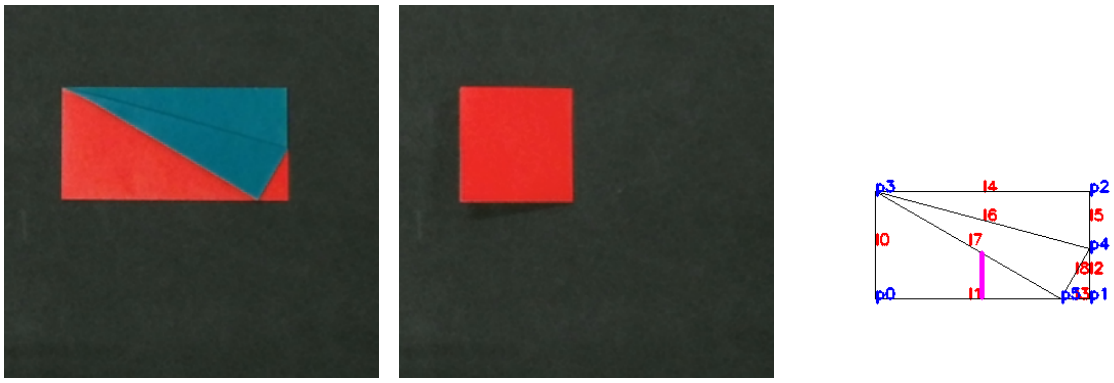



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.19: (22) の折り動作推定結果

表 4.14: (22) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 1 | 0 | |
| 5 | 0 | 3 | 1 |
| 3 | 0 | 1 | |

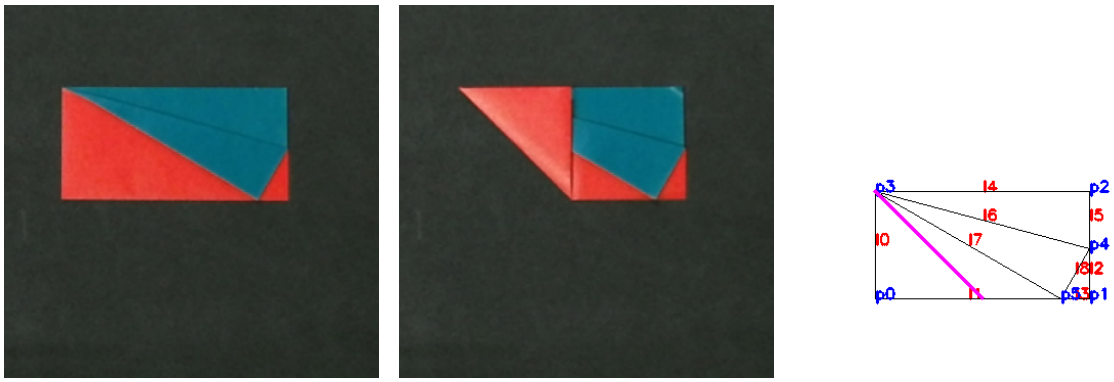



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.20: (23) の折り動作推定結果

表 4.15: (23) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 3 | 2 | |
| 2 | 2 | 3 | |
| 2 | 0 | 1 | |
| 2 | 1 | 0 | |
| 3 | 0 | 2 | |
| 3 | 2 | 0 | |
| 3 | 0 | 5 | |
| 3 | 5 | 0 | |

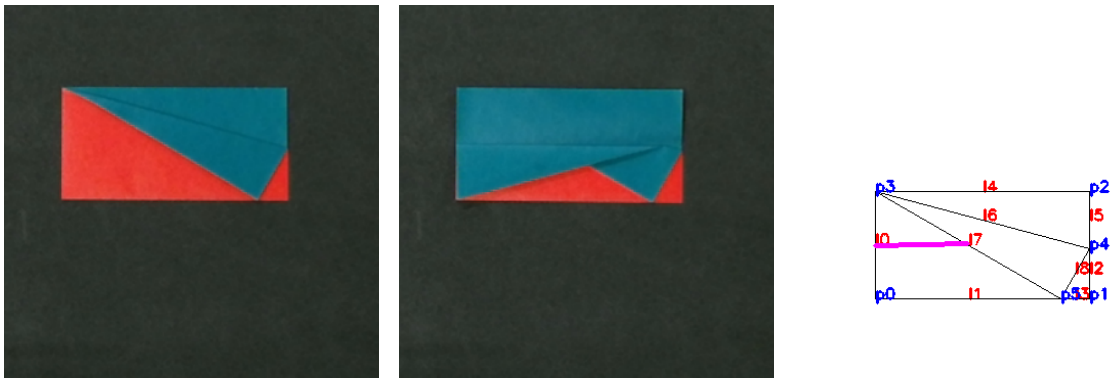



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.21: (24) の折り動作推定結果

表 4.16: (24) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 4 | 0 | |
| 5 | 3 | 0 | 4 |
| 3 | 0 | 4 | |




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

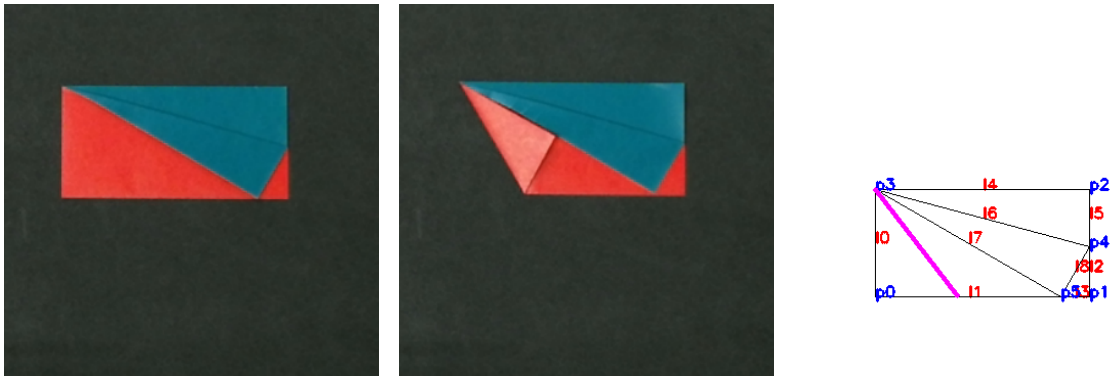
図 4.22: (25) の折り動作推定結果

表 4.17: (25) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 2 | |
| 2 | 2 | 1 | |
| 2 | 3 | 0 | |
| 2 | 0 | 3 | |

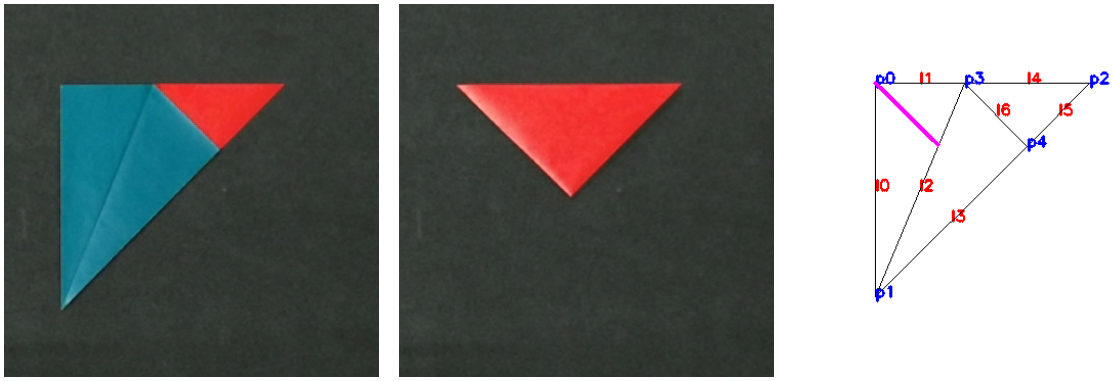
表 4.18: (26) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 5 | 3 | 0 | 6 |
| 3 | 0 | 6 | |
| 3 | 6 | 0 | |




(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.23: (26) の折り動作推定結果

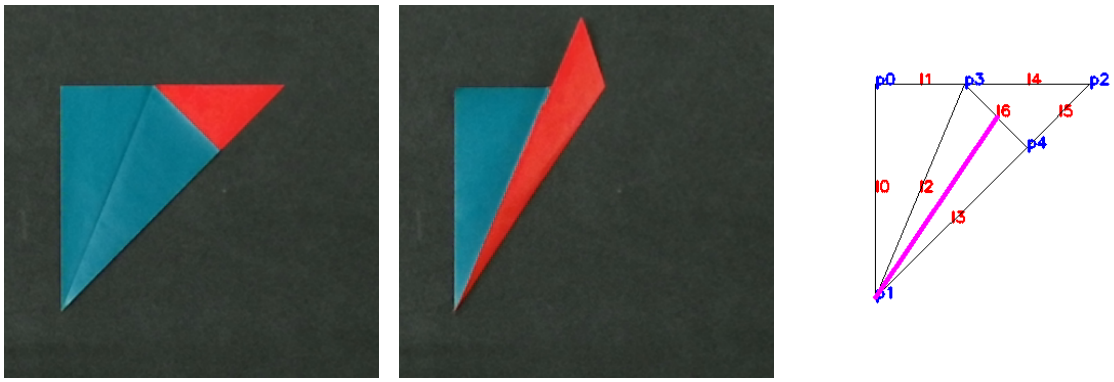



(a) 動作前画像　　(b) 動作後画像　　(c) 検出された折り線

図 4.24: (27) の折り動作推定結果

表 4.19: (27) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 4 | 5 | 5 | |
| 4 | 6 | 5 | |
| 4 | 0 | 5 | |

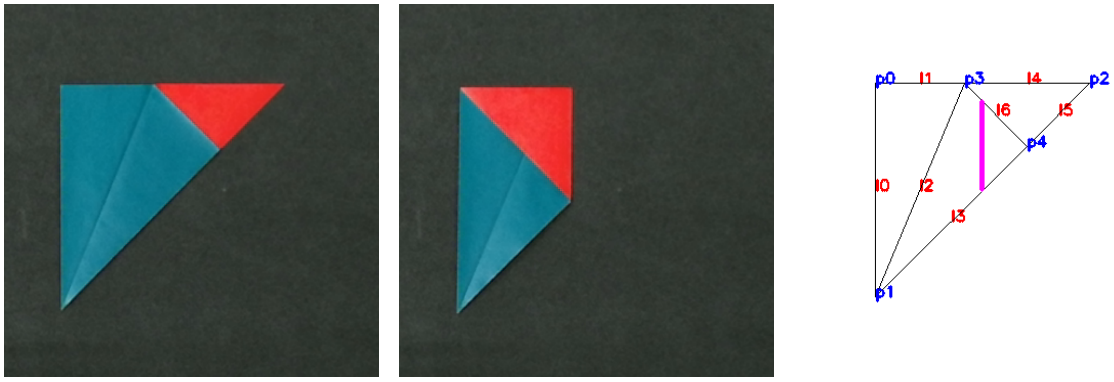



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.25: (28) の折り動作推定結果

表 4.20: (28) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 3 | 2 | |
| 5 | 1 | 3 | 5 |
| 5 | 1 | 4 | 2 |
| 3 | 2 | 3 | |

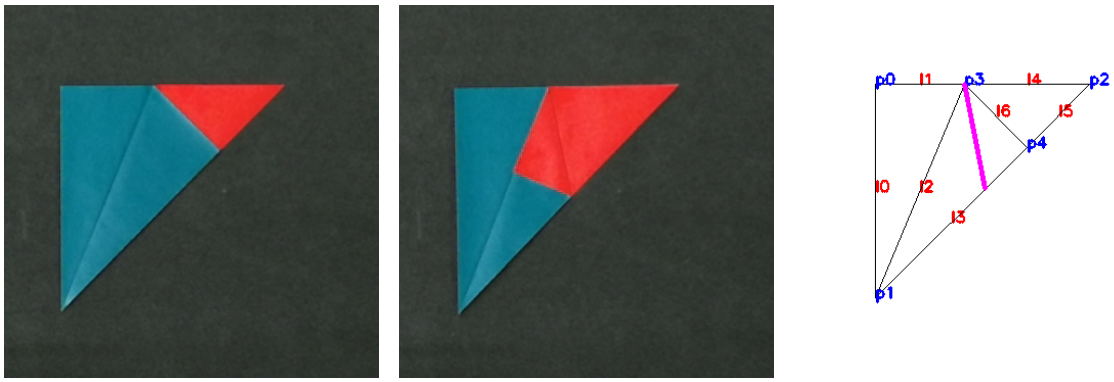



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.26: (29) の折り動作推定結果

表 4.21: (29) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 2 | 0 | |
| 2 | 0 | 2 | |

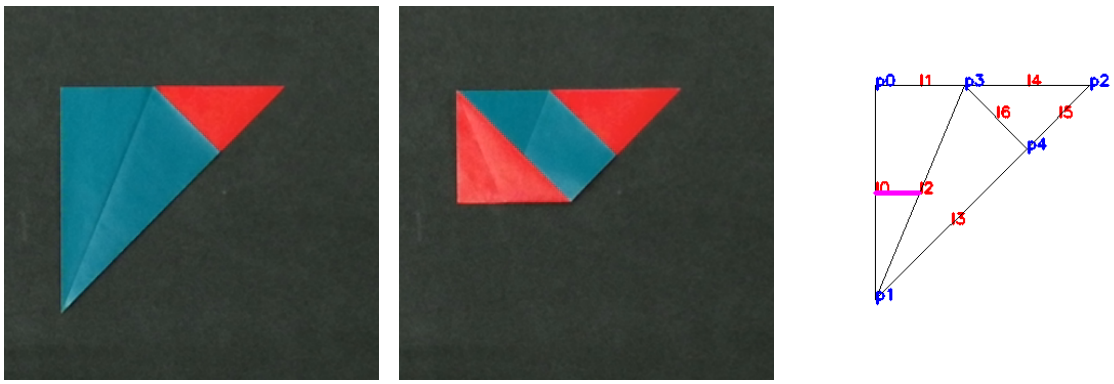



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.27: (30) の折り動作推定結果

表 4.22: (30) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 6 | 2 | |
| 3 | 2 | 6 | |
| 5 | 3 | 4 | 2 |

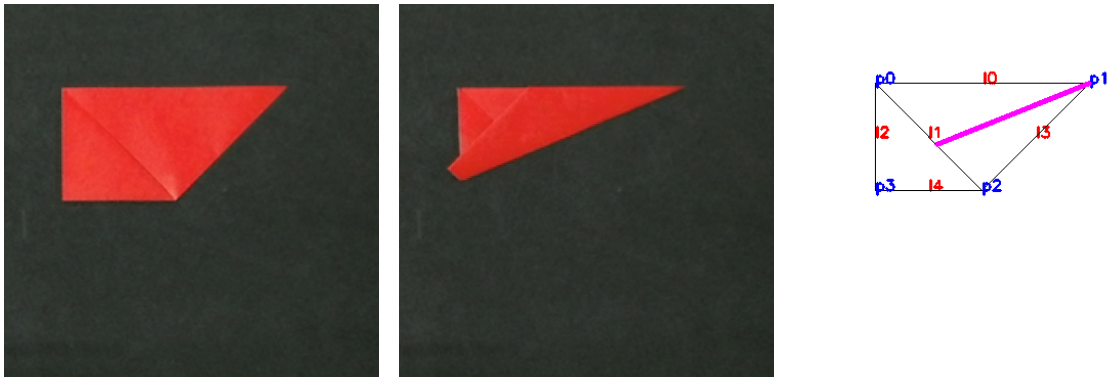



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.28: (31) の折り動作推定結果

表 4.23: (31) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 0 | |
| 2 | 0 | 1 | |




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

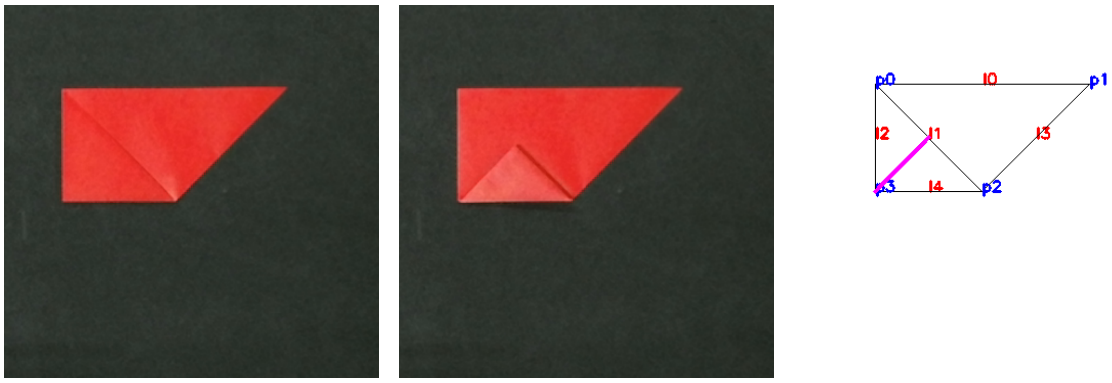
図 4.29: (32) の折り動作推定結果

表 4.24: (32) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 3 | 3 | 0 | |
| 3 | 0 | 3 | |
| 5 | 1 | 2 | 0 |

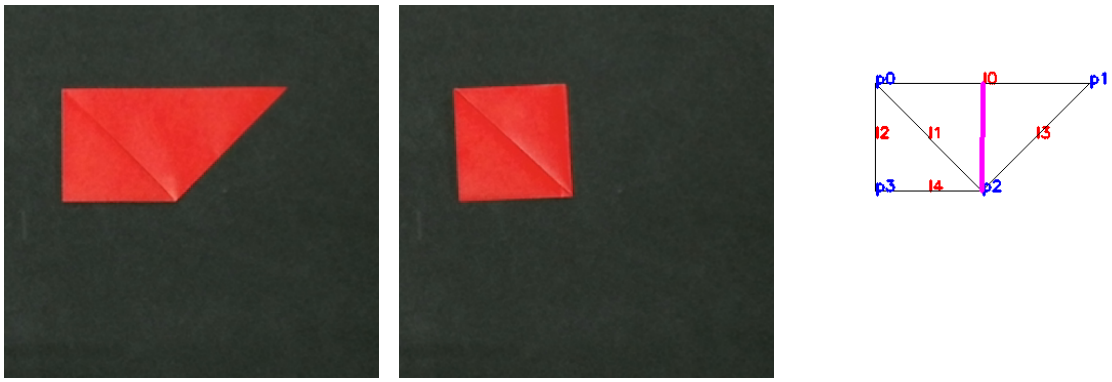
表 4.25: (33) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 2 | 0 | |
| 2 | 0 | 2 | |
| 3 | 2 | 4 | |
| 3 | 4 | 2 | |
| 4 | 3 | 1 | |
| 5 | 3 | 0 | 4 |
| 5 | 3 | 2 | 2 |




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.30: (33) の折り動作推定結果

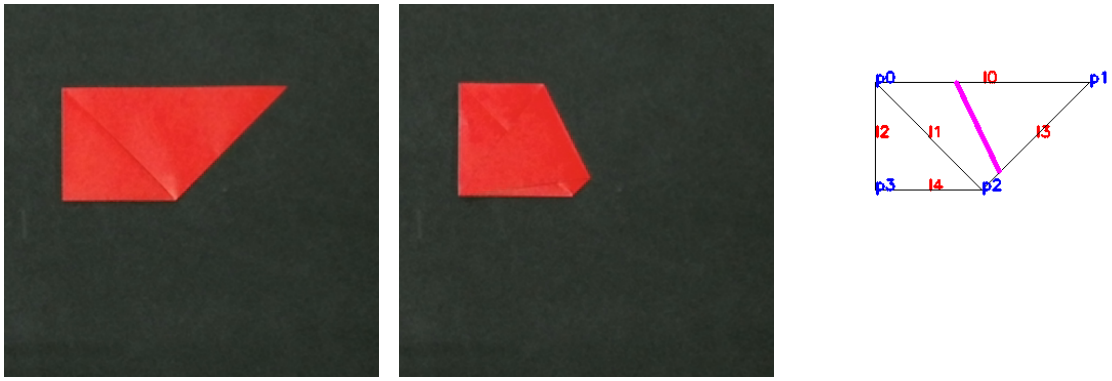



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.31: (34) の折り動作推定結果

表 4.26: (34) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 1 | 0 | |
| 2 | 0 | 1 | |
| 3 | 3 | 1 | |
| 3 | 1 | 3 | |
| 4 | 2 | 0 | |
| 5 | 2 | 0 | 3 |
| 5 | 4 | 4 | 1 |
| 5 | 2 | 1 | 2 |

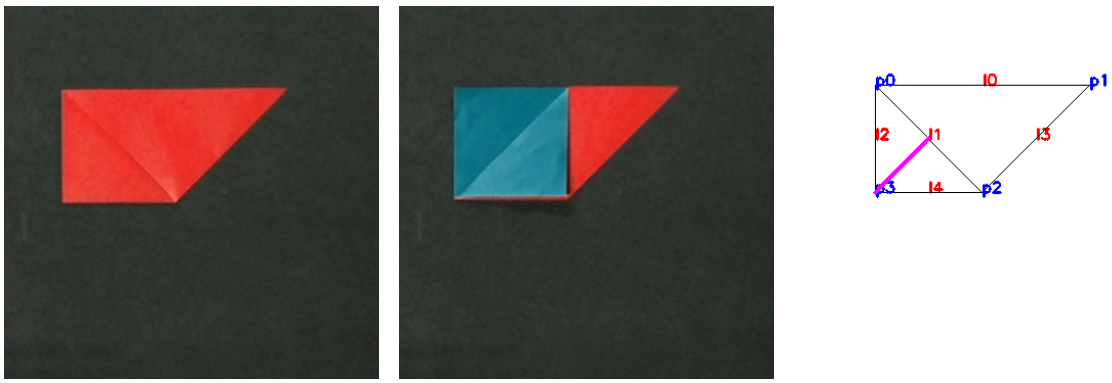



(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.32: (35) の折り動作推定結果

表 4.27: (35) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 3 | | 1 |
| 2 | 1 | 3 | |




(a) 動作前画像　(b) 動作後画像　(c) 検出された折り線

図 4.33: (36) の折り動作推定結果

表 4.28: (36) の操作推定結果

| 公理の番号 | 引数 1 | 引数 2 | 引数 3 |
|---|---|---|---|
| 2 | 2 | 0 | |
| 2 | 0 | 2 | |
| 3 | 4 | 2 | |
| 3 | 2 | 4 | |
| 4 | 3 | 1 | |
| 5 | 3 | 0 | 4 |
| 5 | 3 | 2 | 2 |

# 第5章　まとめ

本研究では，紙を谷折りする作業を対象として，人間によって行われた谷折り操作を認識し，ロボットの作業記述に適した形で記述する手法を提案した．そのために，折り紙作業を逐次的な「操作の連続と見なし」，操作前後の紙の状態と実行された操作の内容を明確にする記述方法を提案し，さらに人間が行う折り操作からその記述に必要なパラメータを推定する手法を提案する．まず，折り紙公理の折り紙の線や点の扱いについて分析し，どのような情報が折り紙公理に基づく作業記述にとって必要か示した．それに基づき折り紙公理で作業を記述するためのデータ構造を定義し，それを用いて折り紙作業を記述する手法を示した．次に，折り線検出に基づく折り操作の認識手法の提案および評価を行った．ここでは，折り操作前後の画像から折り線を画像認識により推定した．その後，行われた操作の種類と折り動作後の折り紙の状態を推定する手法を提案した．提案した手法に対し実験により評価を行い，この手法により二回折りまでの折り作業が推定可能であることを示した．続いて 上述の手法で対応できない折り方に対応する手法を検討した．この手法では，まず折り動作前の折り紙の状態に対し，折り紙公理から記述可能な全ての折り線を列挙する手法を提案した．その際，折り線には適用された公理の情報を保存することで折り線自体に作業としての情報を持たせた．その後，実際の折り動作後の形状に最も近い折り線を選び出し，折り動作後の折り紙の状態を推定する手法を提案した．示した手法に対して実験を行い，三回折りまでの折り作品に対して，認識可能であることを示した．また，認識失敗した例から，このアルゴリズムが点や線の増加により認識率が低下するが示唆されたため，それに対応するための今後のアルゴリズムの改良についての検討を行った．

今後，これをロボットによる観察学習のための手法とすることを目的とし，3回折り以降の折り作業に対しても認識可能であることを示していく．本研究で扱った認識手法であるが，

計算時間が多くかかりすぎることや，複雑な折りに対しては正しい結果が得られない場合が存在することが予想される．そういった要因に折り紙の形状を特徴点として正しく認識できないことや，折り線の候補が膨大になることがあげられる．今後，折り紙の特徴点や構造を正しく得るために，ロボットによるアクティブセンシングなどの必要性が考えられる．また，列挙される折り紙の状態を枝刈りする手法についても検討の余地がある．今回は谷折りしか検討していないため，その他の折り方として存在する山折りや，花弁折り，中割り折り，袋折りなどといった折り方に対応していない．今後そういった多種類の折りに対して認識が可能な手法について検討する必要性も課題として考えられる．

# 謝辞

本研究を行うにあたり，お忙しい中日頃よりご意見ご指導いただきました知能システム学講座 工藤俊亮准教授， 末廣尚士教授，冨沢哲雄助教に感謝いたします．また，日頃よりプログラムの作成等に様々な助言をいただいた，滝澤 優氏に感謝いたします．本研究は， JSPS 科研費 15K16072 の助成を受けたものです．学部生時代に大変お世話になりました長井隆行教授および長井研究室の皆さまに深謝いたします．最後になりますが，学生生活を送るにあたり，お世話になった知能システム学講座の皆さまに深謝いたします．

# 参考文献


[1] Trinh Van Vinh, 冨沢哲雄, 工藤俊亮, 末廣尚士. マニピュレータによる柔軟紐の片手結び. 第 11 回計測自動制御学会システムインテグレーション部門講演会, pp. 2E3–2, 2010.

[2] Trinh Van Vinh, 冨沢哲雄, 工藤俊亮, 末廣尚士. 紐結びのためのマニピュレーションシステム. 第 28 回日本ロボット学会学術講演会, pp. 1P2–1, 2010.

[3] Trinh Van Vinh. 単腕ロボットによる再利用性の高いひも結び動作の実現. 電気通信大学 IS 科 MS 専攻 2010 年度修士論文, 2010.

[4] Masaru Takizawa, Shunsuke Kudoh, and Takashi Suehiro. Method for placing a rope in a target shape and its application to clove hitch. *The 24th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 646–651, 2015.

[5] Masaru Takizawa, Shunsuke Kudoh, and Takashi Suehiro. Rope-placing method for table-top knotting and its application to clove hitch. *The 12th IEEE Transdisciplinary-Oriented Workshop for Emerging Researchers*, 2015.

[6] Naohiro Hayashi, Tetsuo Tomizawa, Takashi Suehiro, and Shunsuke Kudoh. Dual arm robot fabric wrapping operation using target lines. *Int'l Conf. on Robotics and Biomimetics*, pp. 2185–2190, 2014.

[7] Yuji Yamakawa, Akio Namiki, Masatoshi Ishikawa, and Makoto Shimojo. One-handed knotting of a flexible rope with a high-speed multifingered hand having tactile sensors. *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 703–708, 2007.

[8] Yuji Yamakawa, Akio Namiki, and Masatoshi Ishikawa. Motion planning for dynamic knotting of a flexible rope with a high-speed robot arm. *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 49–54, 2010.

[9] 上田, 根木, 吉川. 強化学習を利用した多指ロボットハンドによるページめくり作業の獲得. 第 20 回日本ロボット学会学術講演会予稿集, 2002.

[10] 鴨谷悠介, 田中健太, 横小路泰義. 折り紙作業を題材とした作業解析に基づくロボットハンドの設計と作業実現. 第 24 回日本ロボット学会学術講演会, 2006.

[11] Kenta Tanaka, Yusuke Kamotani, and Yasuyoshi Yokokohji. Origami folding by a robotic hand. *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2540–2547, 2007.

[12] 田中健太, 横小路泰義. 人間の直接教示動作に基づいた折り紙ロボットの動作生成方法. 第 25 回日本ロボット学会学術講演会, 2007.

[13] 大島裕貴, 木原康之, 横小路泰義. 直接教示の容易性と高難易度の折り紙作品の実現を考慮したロボットハンドの設計. 第 12 回システムインテグレーション部門講演会, 2011.

[14] エリック・D・ドメイン, ジョセフ・オルーク. 幾何学的な折りアルゴリズム. 近代科学出版社, pp. 309–321, 2009.

[15] Humiaki Huzita. Tracking deformable objects with point clouds. *2013 IEEE International Conference on. IEEE,*, 2013.

[16] Koshiro Hatori. K's origami: Origami construction.

[17] Robert J. Lang. Origami and geometric constructions.

[18] 鶴田直也, 三谷純, 金森由博, 福井幸男. 折り図作成を支援する手順予測インタフェースと次の手順候補に対するランク付け手法. 第 9 回 NICOGRAPH 春季大会, 2010.

[19] 三谷純. 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. 情報処理学会論文誌, Vol. Vol48, No. No9, pp. 3309–3317, 2007.

[20] 前田雄哉, 冨沢哲雄, 工藤俊亮, 末廣尚士. 折り紙公理に基づいた作業記述-ロボットアームによる折り紙作業実現に向けて-. 計測自動制御学会システムインテグレーション部門講演会, pp. 3B2–3, 2011.

[21] 前田雄哉. 折り紙公理に基づいたロボットアームによる折り紙作業. 電気通信大学 IS 科 MS 専攻 2011 年度修士論文, 2011.

[22] John Schulman, Alex Lee, Jonathan Ho, and Pieter Abbeel. Tracking deformable objects with point clouds. *2013 IEEE International Conference on. IEEE,*, 2013.

[23] M. Hashimoto and T. Ishikawa. Dynamic manipulation of strings for housekeeping robots. *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication*, p. 368?373, 2002.

[24] 須藤克仁, 角所考, 美濃導彦. 現実物体の観測に基づく線状柔軟物体の操作時の形状のモデル化. 情報処理学会論文誌, Vol. 43, No. 12, pp. 239–256, 2002.

[25] 三谷純. 二次元バーコードを用いた紙の折りたたみ構造の認識とそのモデル化. 情報処理学会論文誌, Vol. Vol48, No. No8, pp. 2859–2867, 2007.

[26] Shinya Miyazaki, Takami Yasuda, Shigeki Yokoi, and Junichiro Toriwaki. An origami playing simulator in the virtual space,. *The Journal of Visualization and Computer Animation*, Vol. Vol7, pp. 25–42, 1996.

[27] 古田陽介, 木本晴夫, 三谷純, 福井幸男. マウスによる仮想折り紙の対話的操作のための計算モデルとインタフェース. 情報処理学会論文誌, Vol. Vol48, No. No12, pp. 3658–3669, 2007.

[28] P.J. Besl and N.D Mckay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Analysis and Machine In-telligence*, Vol. 14, No. 2, pp. 239–256, 1992.