平成 15 年度博士論文

計算機による立体紙模型の設計支援に関する研究

鈴木宏正教授

東京大学大学院工学系研究科
精密機械工学専攻

工 07258 三谷純

概要

紙を用いて立体を作成するペーパークラフトやポップアップカードは、その手軽さからホビー分野をはじめとして、建築物のプレゼンテーションへの活用や、ノベルティグッズとしての利用、教育分野での教材としての活用など、幅広い分野で用いられている。しかし、これらは展開図があればハサミと糊を用いて手軽に組み立てられる反面、意図した形状に組み上がる展開図を作成することは難しい。そのため、現在でもこのような立体紙模型の展開図は熟練者の手によって試行錯誤を元に作成されることが多い。そこで立体紙模型の展開図の作成を計算機によって支援することを目的とした研究を行った。本研究では大きく分けて、CG の世界で 3 次元形状を表現する際に一般的に用いられているポリゴンモデル、および面の数が多いメッシュモデルから、その紙模型を作るための展開図生成方法と、ポップアップカードの一つの技法として知られている折り紙建築の設計を支援する方法の 2 つのテーマを扱う。まずポリゴンモデルの展開図作成について、計算機内でのデータの扱い方とアルゴリズムを提案し、展開図の組み立てやすさを評価するためのコスト評価式、およびコスト評価を用いた展開図の作成と、展開図編集のためのインターフェースの提案を行う。続いて、工作するには面の数が多すぎるようなメッシュモデルについて、全体を帯状の領域に分割することで、紙の柔軟性を活かした滑らかな紙模型を作成できるようにする、近似的な展開図の生成手法を提案する。折り紙建築については、ボクセル表現を用いた効率的な設計支援方法と、平面多角形の集合を用いた自由度の高い設計方法、および格子状の紙の組み合わせを用いた 180 度タイプの折り紙建築の設計方法について提案する。本論文で提案するそれぞれの手法について、実際に計算機上にアプリケーションを実装し、それぞれの評価を行った。本手法によって、計算機内で構築されたデータから紙模型用の展開図を迅速に作成できること、および一般のユーザーには難しかった紙模型の設計を計算機によって効率的に支援できることを確かめた。

目次

図目次

表目次

第1章

序論

第 1 章

序論

1.1　はじめに

紙を用いて立体を作成するペーパークラフトやポップアップカードは、その手軽さからホビー分野をはじめとして、建築物のプレゼンテーションへの活用や、ノベルティグッズとしての利用、教育分野での教材としての活用など、幅広い分野で用いられている。人間がモノを知覚するためには、平面の印刷物よりも、実際に手で触れられ、好きな角度から自由に眺められる立体形状の方が優れていることは明らかであり、この立体形状を手軽に作成できる手段として、紙を用いた模型の作成は様々な分野で活用されている。

紙で立体を作る手法の一つであるペーパークラフトは、おそらく誰もが子供のころに体験するものであり、工作用紙とハサミと糊、もしくはそれに代わるものさえあれば手軽に立体模型を作成することができる。また、ポップアップカードの仕組みは、飛び出す絵本やグリーティングカードなどに活用されている。現在では、これらの展開図が書籍やインターネットを通じて入手できるようになっている。

特に近年では、一般家庭にパーソナルコンピュータとプリンタおよびインターネットが広く普及し、展開図をダウンロードして自宅で印刷することが可能になったことで、立体紙模型の作成はますます我々の身近なものになりつつある。企業が自社製品の販促ツールとして、製品や企業キャラクターのペーパークラフトをインターネット上で配布することが、既に一般的に行われている。

しかしながら、紙模型はその展開図の配布や組み立てが手軽な反面、意図した立体を実現するための展開図を作成することは一般に難しく、現在流通しているホビー用の展開図などは、専門技術を持つ設計者によって試行錯誤を経て作成されることが多い。また、建築物のプレゼンテーション用に作られる模型なども、技術者が手作業で図面から展開図を作成し、幾度も試作を繰り返すことが一般的である。インターネットによる展開図の流通とプリンタによる展開図の出力など、紙模型に関する多くの部分はデジタル化されつつあるが、最も上流の展開図の設計ステージにおいては、未だ熟練を要するアナログなままである。この原因の一つは、展開図の作成の難しさに拠るものと考えられる。

この展開図の設計を計算機によって支援することで、意図した形状の展開図を容易に作成できるようになれば、さらにより多くの場での活用を望めると考えられる。

そこで本論文では、計算機によって立体紙模型の設計を支援するための研究を行うものとした。

1.2 立体紙模型の活用分野

紙で作る立体模型は、ホビー分野をはじめ工業の分野に至るまで幅広い領域で活用することができる。本節では、計算機による展開図の設計支援を活用できると考えられる分野について、それぞれの現状と今後の可能性について述べる。特にラピッドプロトタイプについては、紙模型を新たに活用できる分野として詳しく紹介する。

ホビー産業　一般的な認識として、ペーパークラフトはホビーの一つとして捉えることができる。実際、趣味のためのペーパークラフトや折り紙建築の書籍は多数市販されており、また近年ではインターネットの普及に伴い、 Web サイトからダウンロードできる展開図データも非常に多く存在する。しかし、これらの趣味としてのペーパークラフトは、既存の展開図を元に形を組み立てることが主流であり、自分でオリジナルのペーパークラフトを作成して楽しむことはあまり一般的ではない。これは、意図した形状に組みあがる展開図を作成することが困難であることに拠るものと思われる。

ところで、近年では安価な CG ソフトが普及し、趣味の一つとして 3 次元形状を PC で作成することも一般的になりつつある。そこで、計算機内の立体形状のデータから展開図を容易に作成することができれば、オリジナルの立体形状を作成するという、ペーパークラフトの新しい楽しみ方を創出できるものと考えられる。

工業分野　板金による缶や管の作成においては、金属の平板に展開図を作図し、それを曲げ加工して組み立てることが行われる。素材が金属である点で紙模型とは異なるが、展開図から立体を組み立てるという共通点がある。現在、この展開図の作成は、図学の知識を用いて手作業で行われることが多いが、その場合、任意の形状に対する展開図を作図することは非常に難しいため、予め展開図がわかっている形状を組み合わせて立体形状を作成することが多くなされている。例えば、文献 [1] では、板金で管を作成する際に頻出する曲面と曲面の交わりなどを 42 パターン挙げ、それらの展開図を得るための手法を個別に提示している。これらのパターンに含まれる形状は、文献に記載の方法で展開図を得ることができるが、それ以外の形状については独自に作図する必要があるため、自由な形状を板金で作成するのは困難である。

そこで、事前に CAD ソフトなどで設計した形状に対して、計算機で自動的に展開図を生成できるようになれば、缶や管の作成に自由な形を使用できるようになると考えられる。

教材　現在、中学校の数学の授業では、円錐や円柱、および角錐や角柱の幾何の学習において、表面積の求め方の説明に立体の展開図を利用している [2]。また、初等教育においても、立方体とその展開図の関係などの学習が行われている。これらの学習に必要となる立体と展開図の関係の理解には、紙模型が優れた教材に成りうると考えられる。近年では学校教育の場にも PC が普及しつつあるため、計算機で任意の角数の角錐を作成し、その場で展開図を作成する、などの活用方法が考えられる。

販促ツール　車やバイクなどを製造、販売している企業の中には、ホームページ上で自社製品の模型を組み立てられるペーパークラフトを公開している企業が多い。ゲームソフト関連の企業も同様に、多くのホームページで自社製品のゲームに登場するキャラクターのペーパークラフトを公開している。また、建築関連の企業においては、顧客へのプレゼンテーションとして、

家の模型を提示することが多くなされている。これらは、顧客獲得のための販促ツールの一つとしてペーパークラフトを有効に活用している例である。

しかし、これらの展開図も未だ熟練者による試行錯誤で作成されることがほとんどである。近年の工業製品や、 3D ゲームのキャラクターにおいては、その形状が 3D データとして計算機内に存在することが多いため、これらのデータから自動的に展開図を作成できれば、非常に効率的である。特に建築関連の家屋などのプレゼンテーションについては、顧客ごとに異なる模型が必要になるため、これらの展開図の作成が効率化できれば、模型作成のコスト低減に大きく貢献することができる。

グリーティングカード　クリスマスや誕生日などのイベントに贈るグリーティングカードは、多くの人々に日常的に用いられ、様々な趣向を凝らしたカードが市販されている。その中には、ポップアップカードの原理を用いた飛び出すカードも多い。また、一方で個性のあるカードを贈る為に、手作りのカードも人気があるが、飛び出すカードを自分で設計することは容易ではない。そこで、計算機によって折り紙建築の設計を対話的に行い、展開図を自動的に生成することができるようになれば、オリジナルの飛び出すカードを容易に作成できるようになる。

ラピッドプロトタイピング　ラピッドプロトタイピングとは、製品開発において高速 (rapid) に試作品 (prototype) を製造する技術のことをいう。工業分野では、積層造形法や切削などによって 3 次元 CAD で設計したデータから、直接に形状を作成する技術をさすことが多い。これらの手法では、金型等を製造せずに部品を直接製造できるため、コンカレントエンジニアリング等の高速製品開発に必要な技術と考えられている [3]。

しかしこれらは、大掛かりな機械を必要とし、大きな企業や研究所に存在することがほとんどであるため、一般の人々は気軽に扱うことができない。また、一つの試作を作るのに要するコストも高く、製作にかかる時間も数時間から数日かかる場合もある。

一方、紙模型で立体形状を作成する場合には、工作用紙とハサミと糊（もしくはこれらに代わる道具）があればよく、家庭でも手軽に扱うことができる。紙模型には、精度が出ない、壊れやすいなどの問題もあるが、従来の手法ではコストが高いために手が出せなかった領域において、大いに活用できるものと考えられる。現在では計算機内で 3 次元形状を設計することは身近に行われているので、これらの 3 次元形状データから手軽に紙模型用の展開図を作成できるようになれば、従来のラピッドプロトタイピングに比べ、はるかに小さなコストで模型の作成が実現できると考えられる（図 1.1）。

以下では、参考として従来のラピッドプロタイピングの手段の一つとして広く用いられている光造形法と、近年一般家庭への普及を目指した小型モデルが登場しつつある切削機械について紹介する。

光造形法　積層造形法とは、粉体、樹脂、板、紙などの材料を薄い膜状に積層して機械部品を製造する技術で、紫外線により硬化する樹脂を用いた積層造形法は光造形法と呼ばれている。

光造形法は図 1.2に示すように 3 次元形状データ処理とレーザ光学系等による積層装置を用いて実現され、造形の原理と処理のステップの流れは以下のようになる [4]。

1. 3 次元 CAD 等により設計された機械部品モデル（図 1.2(a)）から高さ方向に等間隔で断面の算出を行い、輪郭線よりなるスライスデータを作製する（図 1.2(b)）。

2. スライスデータに基づいて、輪郭内部を塗りつぶすレーザスポットの軌跡を計算し、レーザ走査軌跡データを生成する（図 1.2(c)）。
3. 感光性樹脂を蓄えたタンクに、上下に層厚さの位置決めが可能な可動テーブルを配置し、第一層の厚さの感光性樹脂を塗る。上方から走査軌跡データに従って紫外線レーザ光を走査し、作製しようとする模型の断面形状に樹脂を硬化させる（図 1.2(d)）。
4. テーブルを下げて液面下に硬化物を沈め、次の層を造形するための 1 層の厚さの感光性樹脂が液面と硬化物上面との間にはさまれるようにする（図 1.2(e)）。
5. 次の層のレーザ走査軌跡データによりレーザ光を照射する。物体断面形状に樹脂を硬化させると、下層と結合した硬化層が形成されていく（図 1.2(f)）。
6. 以後 5. を繰り返し、最終層まで硬化させて立体模型の造形が完了する。

光造型法は、複雑な形状でも造形プロセスの大きな変更もなく成形可能であるという特長を有している [5]。しかし、立体データを加工することで加工コストを低減するための研究もなされている [6] が、感光によって樹脂が硬化するのに要する時間も 10 立方センチメートルで十時間程度と長く、また樹脂そのものも高価であり、紙模型のように手軽に扱うことはできない。

切削加工 立体形状を素材から削りだす工法を切削加工という。工業製品の金型製作などの高い精度を要求される工程では、高度な技術が要求されるため、非常に高価な NC 切削機が必要とされるが、近年ではプロトタイプを手軽に作成するための安価な加工機も普及しつつある。安価なものには、平面方向の自由度を持つワークと高さ方向の自由度のみを持つ切削工具によって 2.5 次元の形状のみを作成できるものもある。例えばローランド DG 社製の「3D プロッタ」と呼ばれる加工機 [7] には、 PC 上のソフトで作成した立体データを手軽に切削できる製品がある（図 1.2）。これらは、従来の高価な NC 切削機に比べ工具の自由度と精度が低く、使用できる素材が限られるなどの問題点があるが、低価格で導入できるメリットは大きい。しかしながら、切削には数時間を要し、数十万円という機器の価格は一般家庭で手軽に導入できるものではない。また、これは光造形と共通の課題であるが、得られるのは立体の「形状」のみであり、その質感や重さ、また紙模型では容易に実現可能な表面の模様や色の再現ができないという問題がある。

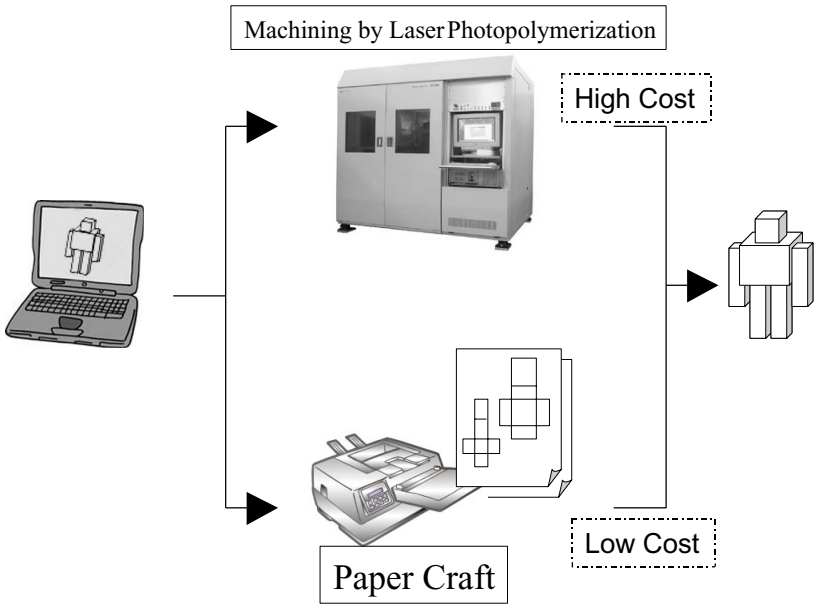


図 1.1: 光造型などの大型機械を用いた試作と紙模型を用いた試作

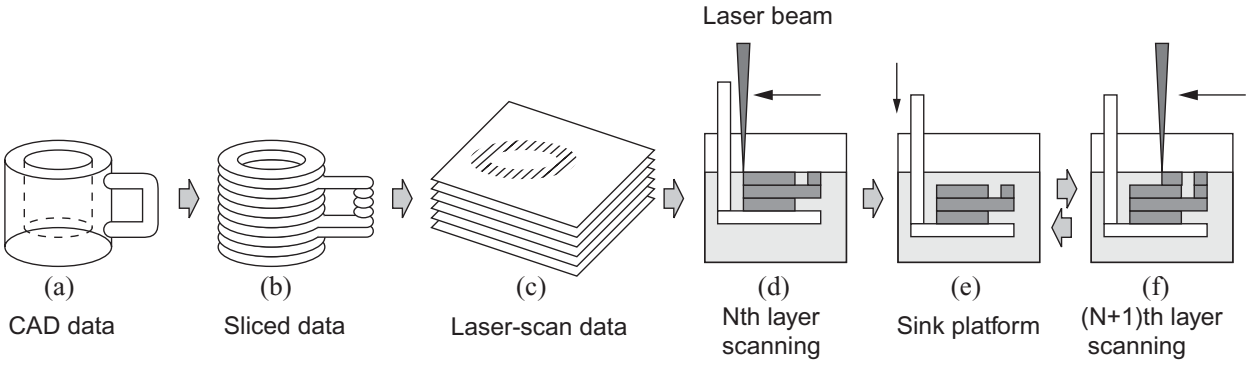
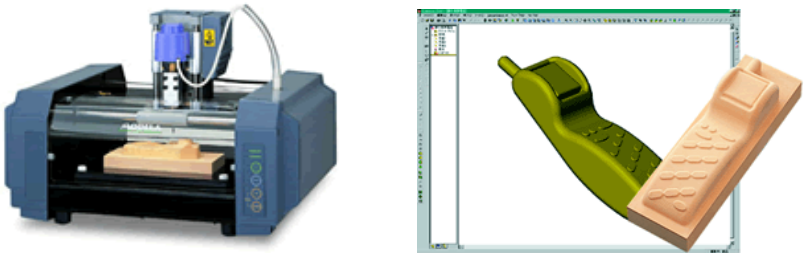


図 1.2: 光造形の流れ



(a) (b)

図 1.3: 3D プロッタ
(a) 3D プロッタ機器 (b) PC で設計したデータと切削された形状
（ローランド DG[7]）

1.3　本研究の目的

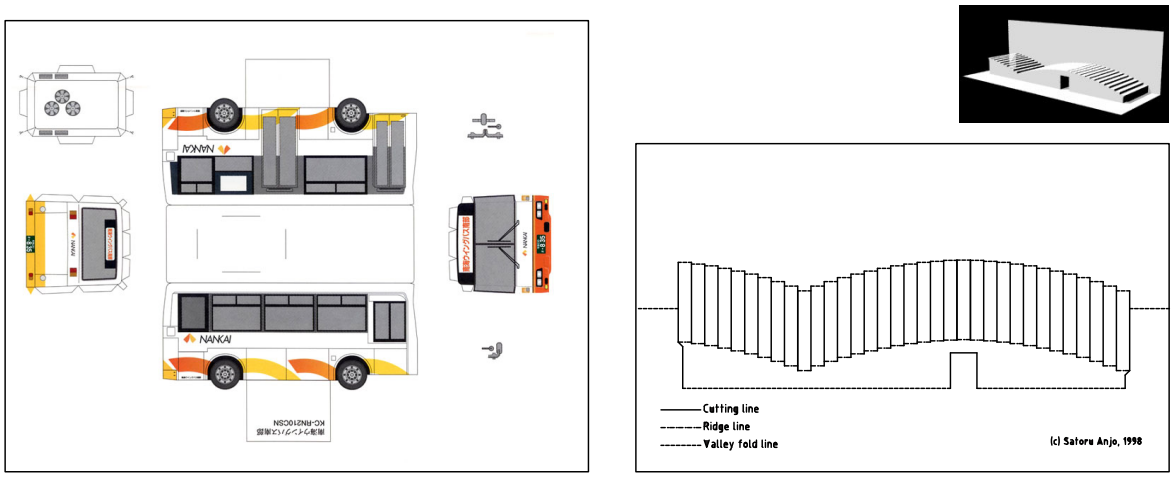
本研究の目的は、計算機によって紙模型の設計を支援することである。

紙で立体を作る紙模型には、**ペーパークラフト**とよばれ広く親しまれているものの他に、開くと形が立ち上がる**ポップアップカード**も存在する。また、日本固有の伝統文化として知られる**折り紙**や、紙を重ねて貼り合わせて形を作る**はりこ**など、その素材と手法によって様々なものが存在する。

本研究では、作成できる形状の自由度とその組み立ての容易さから、ペーパークラフトとポップアップカードタイプのものを研究の対象とする。なお、ポップアップカードタイプの紙模型については、日本で**折り紙建築**と呼ばれ親しまれているものが存在するため、特にこの折り紙建築に着目して、その設計についての研究を行うものとした。このいずれも、立体形状を作るための**展開図**を必要とするため、この展開図を生成するための理論の研究と、それを実現するための技術開発を研究の中心に据える。また、静的に展開図を生成するだけでなく、ユーザーが意図したペーパークラフトおよび折り紙建築を実現できるようなインターフェースの考案も本研究の課題の一つである。

ペーパークラフトの展開図は図 1.4(a) のようなものが一般的で、展開図を切り抜き、紙の折り曲げと貼り合わせで形を作ってゆく。一方、折り紙建築は図 1.4(b) のようなものが一般的で、 1 枚の紙に切り込みと折り曲げのみを入れることで、 90 度に折ったときに形が立ち上がり、二つ折りに折り畳むことができるという特徴がある。

ここで示したようなペーパークラフトと、折り紙建築の展開図の設計を計算機で支援することによって、ペーパークラフト分野における設計のデジタル化に貢献できるものと考えられる。



(a)　(b)

図 1.4: 一般的なペーパークラフトと折り紙建築
(a) バスのペーパークラフトの展開図（電車・バスペーパークラフト本 [8]）
(b) 折り紙建築の展開図。写真は完成図（幕張と建築のアートガイド [9]）

既に述べてきたが、従来のペーパークラフトや折り紙建築は、その形状の設計と展開図の作成は熟練者の手による試行錯誤によって行われている。例えば乗り物や建築物をペーパークラフトで作ろうとした場合、その立面図から手作業で立体を起こし、切り貼りの試行錯誤ののちに最終的な形状の決定を行っている。また、動物のような曲面を持つペーパークラフトを作ろうとした場合には、完成イメージを頭に描きながら紙片の折り曲げと切り貼りを行い、試行錯誤を経てそれらしい形を作ってゆく。ある程度の形が完成したら、今度は逆にそれを切り開き、展開図としてトレースしなおすことで、ペーパークラフト用の展開図を作成している（2.1.2節参照）。折り紙建築の設計も、その独特な形状の制約から、展開図の設計には図学の知識が必要とされ、展開図から実際に形が立ち上がるかどうかを試行錯誤によって確認する必要がある。

これらの手法は、一般の人々には難しく、熟練者であっても手間と時間を要するものである。本研究で提案する手法では、計算機内に構築された立体形状から展開図を生成するため、試行錯誤によって形状を決定するステップをすべて計算機の中で行うことができる。対象とする形状は、実際の紙を用いずにデジタルデータとして扱うため、既存の CG や CAD による 3 次元形状設計と同じ手法で最終形状を作成、確認でき、新しいペーパークラフトや折り紙建築を設計する際のコストを低減できると考えられる（図 1.5）。

なお、本稿で扱うペーパークラフトについては、その形状は既存の CG や CAD ソフトによって容易に設計できるため、計算機で立体形状を作成するための方法については特に議論せず、主にこれらのデータから展開図を作成するための方法を提案する。一方、折り紙建築については、例えば「1 枚の紙から作成されなくてはならない」、「開いたときに形が立ち上がらなくてはならない」などの制約があるため、これらの制約を満たしながら形状を作成できるような、形状設計手法についても議論する。

本研究は上記のように、計算機によって紙模型の設計を支援するための手法を提案するものである。

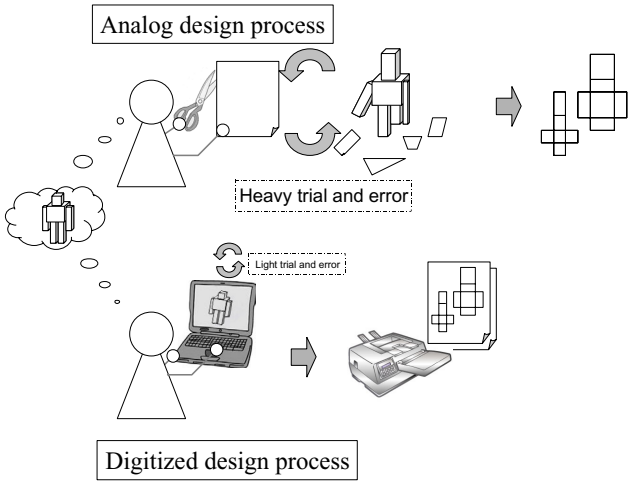


図 1.5: ペーパークラフト設計のデジタル化による試行錯誤に要するコストの低減

1.4 本論文の構成

本論文は本章を含めて全 6 章から構成される。

第 2 章では、紙を用いて立体模型を作成する手法である、折り紙とペーパークラフト、およびポップアップカードについて、それぞれの特徴と、それらの設計を計算機によって支援するための関連研究を紹介する。その後、立体の展開に関する幾何学的な知識をまとめる。

第 3 章では、ポリゴンモデルの展開図を計算機を用いて作成する手法を述べる。まず、計算機で立体を扱うための基礎的な内容を述べ、展開図を作成するためのアルゴリズムや、展開図から紙模型を作成するのに要する組み立てコストの算出方法などの提案、実際に計算機上にアプリケーションを実装した例の紹介などを行う。

第 4 章では、面の数が多く、そのまま展開した場合には工作するのが現実的ではないようなメッシュモデルに対して、近似的な展開図を作成する手法について述べる。ここで提案する手法は一般的なメッシュ簡略化手法とは異なり、帯状の形状の集合でメッシュモデルを近似する点に特徴がある。この手法で作成する展開図では、紙の柔軟性を活かした滑らかな紙模型を作成することができる。また、この手法を実際に面の数が 1 〜 2 万程度のメッシュモデルに適用し、滑らかな曲面部を持つ紙模型を作成した例を紹介する。

第 5 章では、折り紙建築の設計を計算機で支援するための手法を述べる。 90 度に開いたときに形が立ち上がるタイプの折り紙建築について、ボクセルを用いることで設計支援と展開図の作成を効率的に行う方法、および平面多角形の集合を用いることで自由度の高い形状の設計を支援する方法を提案する。また、 180 度型の折り紙建築について、既存のポリゴンモデルから断面形状を取得することで、格子状に形が立ち上がるものを設計するための手法を提案する。それぞれの手法について、その理論と応用についてまとめ、実際にアプリケーションを実装して作成した例題を紹介する。

最後に、第 6 章では本研究の結果によって得られる結論および今後の展望についてまとめる。

第 2 章

紙による立体の表現と展開

第 2 章

紙による立体の表現と展開

本章では、紙を用いて立体模型を作成する手法について一般的に知られている、折り紙とペーパークラフト、およびポップアップカードについて、それぞれの手法の特徴と従来の研究などについてまとめる。特にペーパークラフトについては、近年のインターネットの普及に伴うデジタル化の様子と従来の手法の比較を行う。その後、一般的なペーパークラフト用の展開図を計算機で作成することを考える際に必要になる基礎知識として、立体の展開に関する内容についてまとめる。

2.1　紙を用いた立体の表現手法

我々の生活の中で紙は非常に身近な素材であり、紙を用いて立体を作成する手法は「折り紙」や「ペーパークラフト」などホビーの一つとして親しまれている。また、開くと飛び出すカードとして知られている「ポップアップカード」や「折り紙建築」なども紙によって立体を表現する一つの技法であり、グリーティングカードや飛び出す絵本などに用いられている。本節では、これらの紙を用いた立体の表現手法について紹介する。

2.1.1　折り紙

折り紙の歴史は古く、奈良、平安の昔から貴族社会における儀礼や祭りの中で白紙を「祭忌用」に折ったり切ったり、結んだり、畳んだりしたのが始めと言われている [10]。江戸時代、紙という素材が増産され印刷出版も盛んになると、上流家庭から新興の町人文化、そして一般庶民へと折紙が引き継がれ普及していった。

近年では子供から大人まで幅広く親しまれ、様々な形の折り方を説明する折り紙教本が多数出版されている。また、「日本折り紙学会 [11]」では学術的な観点からも折り紙の研究がなされ、機関紙を通じて折り紙に関する議論が行われている。

また、計算機を用いて折り紙の作成をシミュレーションする研究もなされている。宮崎らは、折り紙の基本的な操作を「折り返し」「折り曲げ」「折り込み」の 3 つに分類し、それを元に 1 枚の紙から立体を作成する様子を計算機でシミュレートするシステムを構築した [12]。

(a)　(b)　(c)　(d)

図 2.1: 計算機による折り紙のシミュレーション
(a) 折り返し (b) 折り曲げ (c) 折り込み (d) 基本的な折り方の繰り返しで作成した鶴
(宮崎 [12])

折り紙の研究は特に日本国内においては広く行われているが、形状を精度良く表現するという工学的な目的よりも、数学や幾何学に関する純粋な好奇心や、厳しい制約の中で形を表現する技法のチャレンジ、美しさに重みを置く芸術的な観点からのアプローチが多いように思われる。 1 枚の正方形から折り曲げだけで形を作る、という制約は非常に厳しく、任意の形を作るのは難しいため、 CG や CAD によって計算機内に表現された立体の模型を作成する手段に用いるのは難しい。

2.1.2 ペーパークラフト

ペーパークラフトは折り紙と異なり、紙の折り曲げと共に、切り抜きや貼り合わせを行って立体模型を作成するため、作成できる形状の自由度が非常に高い。素材が紙であるため、目的の立体形状が平面に展開された展開図が最初に与えられるのが一般的である。展開図には、切り取るべき「切断線」と折り曲げるべき「折れ線」が存在する。また、折れ線には凸に折り曲げる「山折り線」と凹に折り曲げる「谷折り線」があり、破線や一点鎖線などの線種で区別することが多い。また、貼り合わせを行う辺同士は、糊などの接着剤による貼り合わせが容易になるように、一方に「のりしろ」をつけることが多い（図 2.2）。

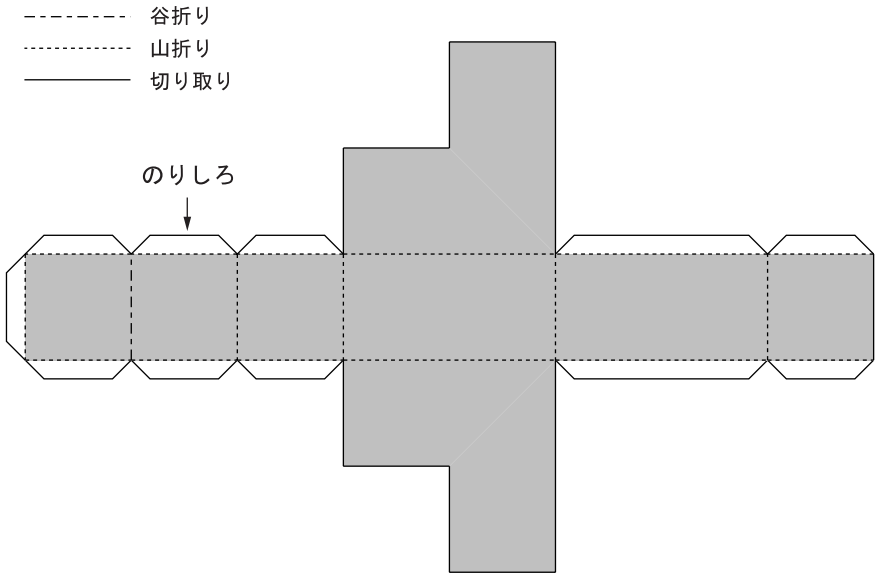


図 2.2: 一般的なペーパークラフトの展開図例

従来のペーパークラフト

ペーパークラフトの展開図の編集や出力などは、パソコンの普及によってデジタル化されつつあるが、その形状設計および展開図の作成における初期段階は未だデジタル化されておらず、技術者の手によって実際に紙とカッターを用いた試行錯誤によって行われることが多い。幾何的な形状特徴を持つ工業製品などは、CAD によって出力された図面を元に形を作ってゆくことが行われるが、動物などの対象物については、まさに職人技の発揮される領域であり、設計者の感性に基づいた形状作成の試行が繰り返し行われる。

ここでは、ペーパークラフト作家として著名な、ごとうけい氏 [13] の作品製作の手順を紹介する（図 2.3）。

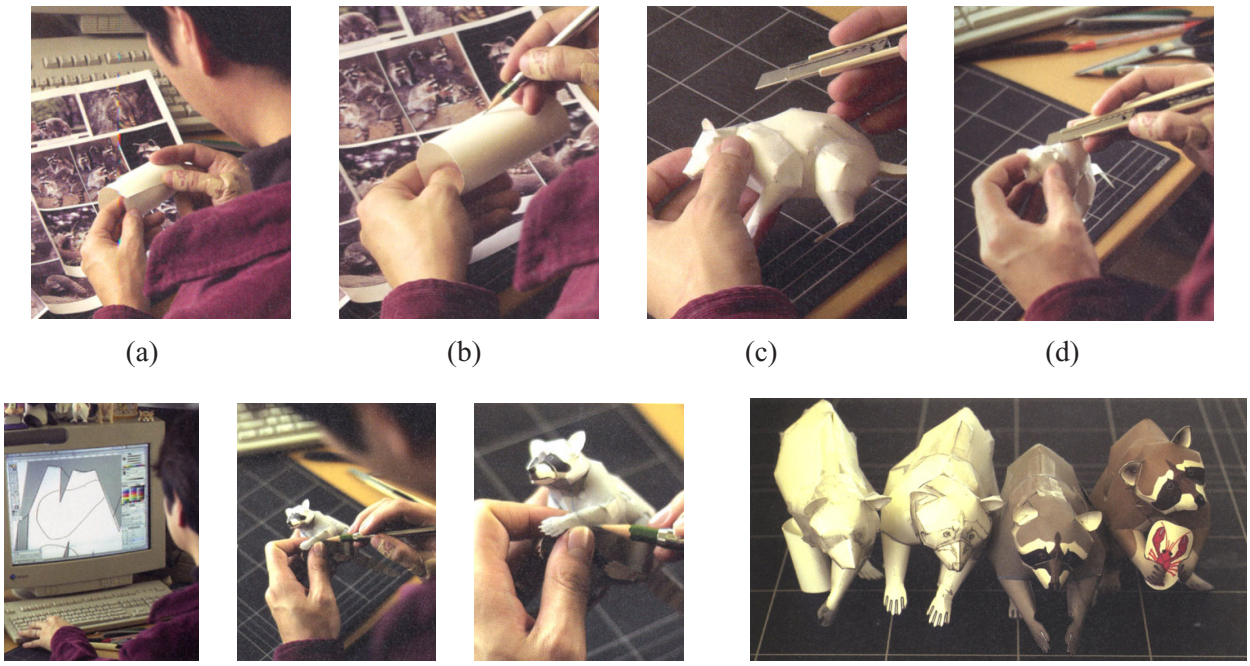
1. **形の作成**（図 2.3(a)(b)）: 最初に平面のデッサンを行う場合があるが、頭の中のイメージを直接紙で作ってゆく。はじめに紙を丸め、それを胴体に見立てて背骨、肩などという各部の場所を定めてはハサミで切り込みを入れ、セロハンテープで留めることを行う。さながら立体によるデッサンである。

2. **展開**（図 2.3(c)(d)）：できあがった立体を切り開いてゆく。開いたものが展開図の原型となる。

3. **展開図のスキャンとトレース**（図 2.3(e)）：展開図の原型をスキャンし、パソコンに取り込んだ後、トレースと着色を行う。

4. **プリントと試作**（図 2.3(f)(g)）：パソコンで作成した展開図をプリントアウトし、組み立てを行う。うまく合わないところなどを確認し、パソコンの画面で展開図の修正を行う。その後、再度プリントアウトして組み立てを行う。

5. **完成**（図 2.3(h)）：上下左右から形状を確認し、納得のゆくものができた時点で完成。試作は 1 つの作品につき 10 回ほど繰り返すという。



(a) (b) (c) (d)

(e) (f) (g) (h)

図 2.3: ペーパークラフト用の展開図作成の手順
(a), (b) 形の作成 (c), (d) 展開 (e) 展開図のスキャンとトレース
(f), (g) プリントと試作 (h) 完成
（デジタルペーパークラフト [14]）

また、ペーパークラフトを作成する立場から見たときは、従来は書店で展開図の掲載された書籍を購入し、それを切り取って組み立てていたため、展開図は 1 つしかなく失敗が許されなかった。やり直しができるように展開図を工作用紙に複写する場合には（筆者も幼いころに行っていた手法であるが）、一度トレーシングペーパーに展開図を写し取り、それを工作用紙の上におき、間にカーボン紙を挟んで、再度展開図を上からトレースすることを行っていた（図 2.4）。このように、工作をするための作業が非常に手間のかかるものであった。

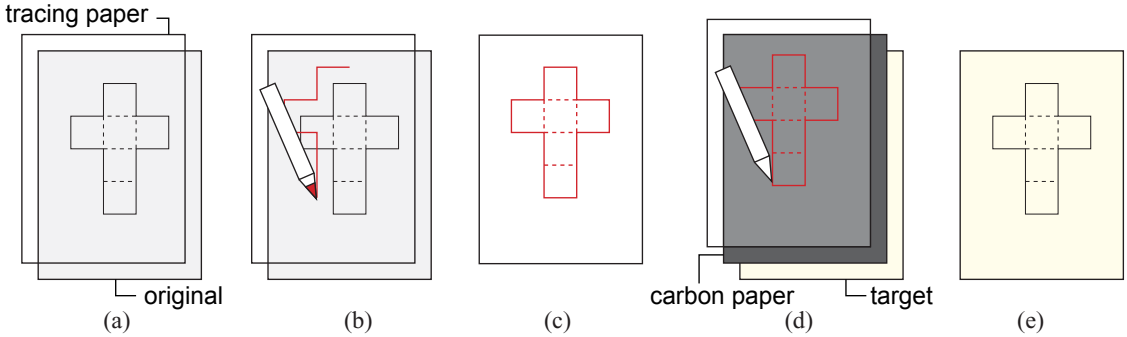


図 2.4: カーボン紙を用いた展開図の複写
(a) オリジナルの展開図にトレーシングペーパーを載せる (b), (c) 展開図をトレースする
(d) 工作用紙の上にカーボン紙とトレーシングペーパーを載せ、再度トレースする
(e) 複写の完了

デジタル化されつつあるペーパークラフト

近年では一般家庭にパーソナルコンピュータとプリンタ、およびインターネットが普及しつつある。このような環境において、従来は展開図の入手から実際の工作まで極めてアナログな作業であったペーパークラフトが、多くの面でデジタル化され、手軽に扱えるようになってきた。現在ではインターネット上に多数の展開図が公開され、展開図が入手可能なサイトのリンク集 [15] も存在するため、 Web サイトから好きなものをダウンロードして入手することができる。これを自宅のプリンタで印刷すれば、工作に失敗しても手軽に何度でも挑戦でき、またスケールを自由に変えて大きさの異なる作品を複数作成することも可能である。さらには、カッティングプロッタを用いることで、展開図の切り出しを自動化することも実現されている [16] （図 2.5）。

図 2.5: カッティングプロッタを用いた展開図の切り出し
（グラフテック株式会社 [16]）

このような状況において、紙模型はより我々の身近なものとなりつつある。紙模型を作成するペーパークラフトは、ホビーの一つとしてはもちろん、例えば建築物のプレゼンテーションツールや、中学・高校の数学教材、または製品の宣伝広告などの場に活用が可能である。

しかしながら、ペーパークラフトの元となる展開図の設計は未だデジタル化されておらず、熟練者の手による試行錯誤によって行われていることが多い。そのため、一般のユーザーがオ

リジナルのペーパークラフトを作成することは難しく、既存の展開図を組み立てることに留まることがほとんどである。従って、この展開図の作成が手軽に行えるようになれば、ペーパークラフトの活用の場はさらに拡大する余地があると考えられる。

従来の CAD にも、機能の一部として展開図の生成が実装されていることがあるが、これらは多面体を展開するだけの簡易的なものが多く、のりしろの生成やテクスチャの適用、展開後の編集機能やより工作しやすい展開図を作るための工夫など、本論文の 3章と 4章で述べるようなペーパークラフトを意識した工夫を施したものは見られない。

また、特にペーパークラフトの設計を計算機で支援することに着目した研究はあまりなされていないが、米村ら [17] は箱型の部品を複数組み合わせて立体を作成する手法のペーパークラフトを提案した。この手法では、平面上に入力された多角形をスイープして作成される箱を、計算機の画面でレイアウトして形を作ってゆく。最終的には、それぞれの箱の展開図と、貼り合わせ位置がわかるようなマークを出力することで、計算機で作成した形を紙模型で実現できる。この手法は、計算機での形の作成と、展開図の生成が容易であり、また組み立ても箱の貼り合わせで済むというメリットがあるが、その一方で作成できる形が箱型の集合に限定されるため、従来のペーパークラフトに見られるような、自由度の高い形を作ることはことはできないという問題がある。

和田ら [18] は、我々が本研究で提案する手法と非常に似た方法で、工作しやすい展開図の生成を意識した展開図の作成手法を提案している。また、ポリゴンで表現された形状を展開して紙模型を作成する際に、形状の中から円錐で近似できる箇所を抽出し、その部位を円錐に置き換えて展開図を生成する手法の提案も行っている。

Elber[19] は、パラメトリック曲面を円錐面や円柱面などの可展面の集合で近似することで、ペーパークラフト用の展開図を作成することを提案した。これは CAD システムの一部に組み込まれ、これによって生成された Utah Teapot（CG の世界ではテストデータとしてよく用いられる有名な形状データ）の展開図が SIGGRAPH では 25 周年を記念して公開され、この展開図は現在でも SIGGRAPH のサイトからダウンロードできる [20]。

2.1.3 ポップアップカードと折り紙建築

ポップアップカードとは、畳まれたカードを開くと立体が立ち上がるものであり、「立体を折り畳める」という性質と「立体が立ち上がる」という性質を持ち合わせている。ポップアップカードは、コンパクトに折り畳めるという特徴を持ちながら立体形状を直接表現できるため、グリーティングカードや飛び出す絵本などに広く用いられている。

ポップアップカードの中の一つの技法に折り紙建築があり、これは茶谷氏が建築物を表現するために用いた 90 度型のものに対して命名したものである [21]。折り紙建築の明確な定義は特に存在しないが、海外を中心に普及している一般的なポップアップカードには 180 度に開いたときに形が立ち上がるものが多いのに対して、主に日本で普及している折り紙建築は 90 度に開いたときに形が立ち上がるものが多いという傾向がある。また、一般的なポップアップカードは複数のパーツを貼り合わせて形を作ることが多いが、折り紙建築では 1 枚の紙から切り込みと折り曲げだけで形を作ることが多い。

ポップアップカードは一般的なペーパークラフトと比較すると作成できる形状の制約が大きく、従来は熟練者の手による試行錯誤で設計が行われてきた。実際に組み立てるまで正しくカードを開閉できるか確認できないため、この試行錯誤のコストは非常に大きい。また、ポップアップカードの中でも特に折り紙建築の場合は「1 枚の紙で作る」という制約があるため、さらに自由な形を作るのは難しくなる。

ポップアップカード

ポップアップカードは、平面に畳まれたカードから立体が立ち上がるというユニークな特徴から、世界中で親しまれ、グリーティングカードや子供向けの本などに活用されている。

また、これらの設計を計算機で支援する研究もいくつか行われている。 Lee ら [22] は、図 2.6に示すような V-Fold と呼ばれる形状について、カードが開いたときに立ち上がるための幾何的な制約と開閉時の軌跡を数学的に明らかにし、これらを組み合わせた形を計算機上でシミュレートした。 Glassner[23][24] は、 Lee と同様 V-Fold のものを中心に、単純なポップアップカードを対話的に作成し、その開閉の様子をアニメーション表示する手法を提案している。

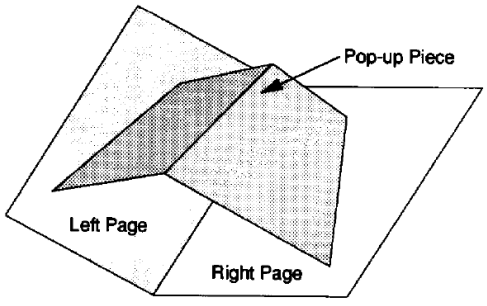


図 2.6: V-fold 形式の単純なポップアップカードの例
（Lee[22]）

折り紙建築

折り紙建築の設計に関する内容は文献 [21] に詳しく、図学的なアプローチで展開図を作図する手法について述べられている。また、折り紙建築は 1 つのホビーとして扱われることも多く、様々な展開図を収録した型紙集が出版されている [25][26]。

折り紙建築を計算機で扱う研究は今までほとんどなされていないが、茶谷らは折り紙建築を CG で表示する例を紹介している [27]。しかし、これは予め形状データが組み込まれた、特定の形状に特化されたプログラムを元に例題を表示するためのものであり、自由に設計したものを扱うことはできない（図 2.7）。先に述べた Glassner[23][24] の手法は、 180 度に開いたときに形が立ち上がる V-Fold のタイプのみを扱っているため、折り紙建築の設計に用いることはできない。このように、折り紙建築の設計を計算機で支援するための研究はほとんどなされておらず、現在では試行錯誤を伴う手作業で設計がなされている。この設計作業を計算機で支援することができれば、オリジナルのグリーティングカードの作成や、ノベルティグッズの作成などに活かす事ができると考えられる。

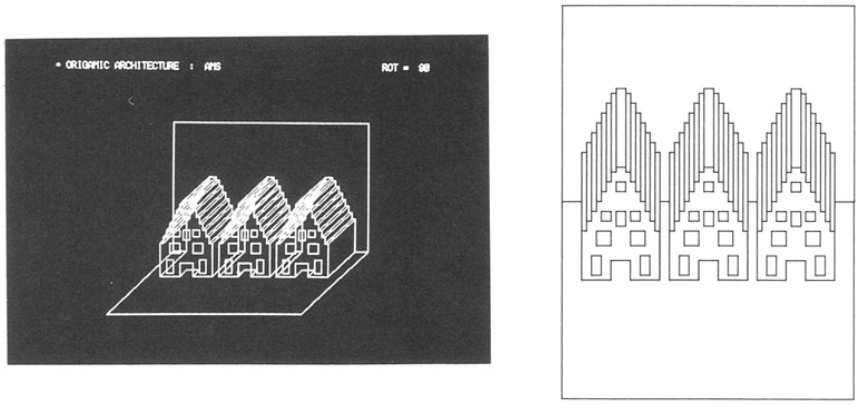


図 2.7: CG による折り紙建築の表示
（実例パソコン 折り紙建築と折り紙 [27]）

2.2 立体の展開

曲面および平面の集合で表現された立体の表面を、剛体変換によって一平面に移した形を、その立体の展開図という。与えられた立体から展開図を得る手法は、古くから図形科学の分野で研究されている。

ところで、剛体変換では曲面のガウス曲率は変わらず、平面のガウス曲率はゼロであるから、平面上に剛体変換できる曲面、すなわち展開可能な曲面はガウス曲率がゼロである曲面に限られる。このように、全ての曲面が平面に展開できるわけではなく、特に平面に展開可能な面を**可展面**と呼ぶ。

曲面は図 2.8のように、その性質によって展開可能なものとそうでないものに分類することができる [28]。

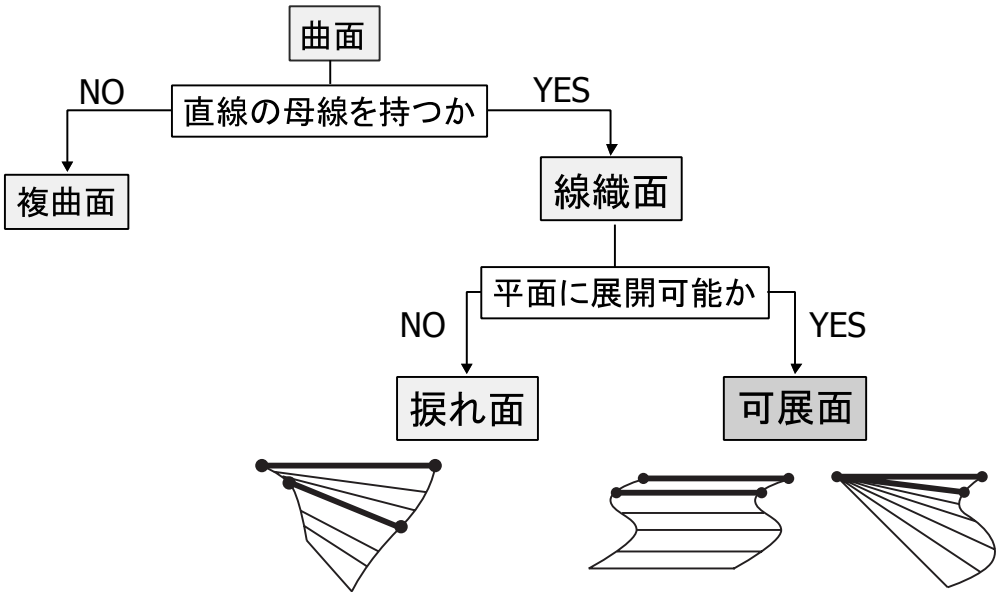


図 2.8: 曲面の分類

まず母線（曲面を構成する線）が直線か否かに注目すると、母線に直線が無い場合、すなわち曲線でしか構成出来ない場合これを**複曲面**と言う。球や多くの回転体も複曲面に含まれるが、それ以外にも無数の複曲面があり、これらは平面に展開することができない。

一方、直線母線を持つ場合を**線織面**という。線織面は、面を伸び縮みさせずに平面上に展開出来るか否かで、さらに分類できる。具体的には、導線沿って近接する 2 つの直線母線の位置関係によって、次のように分類される。

- 交わる → 可展面
- 平行 → 可展面
- 捩れ関係 → 捩じれ面

平面に展開できるものが**可展面**であり、**捩じれ面**は平面に展開することができない。

可展面には、多面体、錐面、柱面、接線曲面、 4 つがあり、それぞれ次のようなものである [29]。

多面体 平面図形の多角形を構成要素にもつ。構成要素の多角形がすべて合同形のものは、正 4 面体、正 6 面体、正 8 面体、正 12 面体、正 20 面体、の 5 つに限られる。多面体の構成要素は平面であるから、その上の任意点を通る直線母線は無数に存在する。

錐面 1 つの平面曲線と別に与えた頂点とを結ぶ直線群で作る曲面で、円錐が含まれる。この平面曲線に底円を持つ錐面には直円錐と斜円錐とがある。

柱面 1 つの平面曲線を通る平行直線群で作る曲面で、円柱が含まれる。頂点が無限遠方にある錐面に相当するため、錐面と共通の性質が多い。

接線曲面 1 つの空間曲線にそった接線群で作る曲面。

可展面は、その名の示すとおり平面に展開できるが、それ以外の曲面を展開する場合には、この可展面の集合で形を近似することで、近似的な展開を行うことができる。

可展面でない曲面を円錐面や円柱面などの可展面の一部の集合で近似して展開図を作成する研究は Elber[19]、 Pottmann and Farin[30]、 Hoscheck[31]、 Simon[32] らによって多くなされている。これらのアプローチは B-spline 曲面や rational Bèzier 曲面などのパラメトリック曲面を円錐面や円柱面などの可展面の集合で近似するものである。

しかし可展面の集合からなる形状も、多面体以外の形状については、その展開図を厳密に作成することはそれほど容易ではない。特に複数の曲面が交わる場合などは、解析的に展開図を得ることは難しくなる。 Elber[19] の手法では、可展面の集合で近似したものを、改めて三角形面の集合で近似しなおして、それを平面に展開することを行っている。

そのため、任意の曲面の近似的な展開図を生成する手法としては、最終的には平面多角形の集合で近似することが現実的なアプローチであると考えられる。平面多角形から構成される多面体は、 CG の世界ではポリゴンモデルと呼ばれ、曲面を近似表現する際にもよく用いられている。

本研究では、計算機内に構築された 3 次元形状を元に、ペーパークラフト用の展開図を作成する際には、このような平面多角形の集合によって表現される形状の展開手法について考慮するものとする。面の数が少ないポリゴンモデルは、そのまま平面に展開し組み立てることができる。一方で、メッシュモデルと呼ばれるような、面の数が非常に大きな形状は、そのままでは工作することが現実的ではないため、元の形状特徴を維持しながら工作の手間を軽減する工夫が必要となる。本研究では STRIP と呼ぶ、一続きの三角形の帯の集合で形を近似することで、この問題を解決する方法を提案する。最終的に形状を展開が容易な三角形の集合で近似するという点においては、従来の研究と同じアプローチであるが、従来の研究では円錐面や円柱面などの可展面を対象としていたのに対し、本研究では細かい三角形によって表現された形状を対象としている点が異なる。

第 3 章

ポリゴンモデルの展開図作成手法

第 3 章

ポリゴンモデルの展開図作成手法

本章では、ポリゴンモデルの展開図を計算機を用いて作成する手法を述べる。まず、計算機によって 3 次元形状を表現する手法について概説し、その手法の 1 つであるポリゴンモデルを扱うための基礎的な内容を紹介する。その後、ポリゴンモデルを展開するためのアルゴリズムを提案し、それを計算機で実装する際に気をつけるべき点などを述べる。また、一つのポリゴンモデルに対して、その展開図は何通りも存在するため、どのような展開図が工作に適した展開図であるかを判断するためのコスト評価方法を提案し、組み立てやすさを考慮した展開図の作成手法について述べる。続いて、得られた展開図をユーザーが対話的に編集するためのインターフェースや、工作を支援するための機能を提案し、それを計算機上に実装したアプリケーションの例の紹介を行う。

3.1 計算機による 3 次元形状の表現

3 次元形状を計算機で取り扱う研究は、 CAD や CG の分野で古くからなされている。近年ではこれらの技術の進歩により、単純な設計やモデル表示だけでなく、そのデータを効率よく加工する技術や、有限要素解析への活用など、応用研究が広く行われるようになってきている。計算機で 3 次元形状を扱う場合には、それを扱う目的に応じて、形状をどのように計算機内で表現するかが重要である。本節では、この 3 次元形状モデルの表現方法と、特にその中の一つの多面体モデル（ポリゴンモデル）による表現についてまとめる。

3.1.1 3 次元形状モデルの表現方法

計算機内での 3 次元形状モデルを表現する方法は、大きく次のように分類することができる。

- **ワイヤーフレームモデル**

  ワイヤーフレームモデルとは、稜線と頂点の座標で 3 次元形状を表現するものである（図 3.1(a)）。稜線は視点と終点の座標値をもち、直線や円弧をはじめ、ベジェ曲線や B スプライン曲線などの自由曲線で表現されることもある。しかしワイヤーフレームモデルは形状の面や中身についての情報をもたないため、立体を正確に表現する事はできない。

- **サーフェスモデル**

  サーフェスモデルとは、ワイヤーフレームモデルのデータに加えて面のデータも持つものである（図 3.1(b)）。立体の表面のデータはあるが、中身についての情報を持たないため、いわば中空のモデルである。面はそれを構成する稜線のデータを位相的な構造として持つ。稜線が直線である場合、面の形状は単純な多角形で表現されるが、それ以外にも球面、円柱面、楕円体面、円錐面といった 2 次曲面の他に、 Coons 曲面、ベジェ曲面、 B スプライン曲面、 NURBS(non-uniform rational B-splines) 曲面などの自由曲面が考案されている。

- **ソリッドモデル**

  ソリッドモデルとは、定義される立体の内部についての情報を持つものである（図 3.1(c)）。ワイヤーフレームモデルとサーフェスモデルは 3 次元形状を線あるいは面の集合体として表現するのに対し、表現された 3 次元形状の内部の情報を持っているため、 3 次元立体を完全に表現でき、論理集合演算による立体の和や差を求めることができる。特に閉じた面の集合でソリッドを表現するものを境界表現モデルという。

- **ボクセルモデル**

  上記のワイヤフレームモデル、サーフェスモデル、およびソリッドモデルが連続値を扱うのに対し、ボクセルモデルは空間を格子状に区切ることで、立体を単位立方体で量子化された離散値の集合で表現する、空間占有法の一つである（図 3.2）。空間の各格子について、立体の内外を識別する 1 ビットの値を保持することで、形状の表現が行われるため、データ構造は極めてシンプルであり、ソリッドモデル同様に物体の内部も完全に表現することができる。その一方で、解像度を高めるとデータサイズが極度に大きく

なり、計算機資源の乏しい環境では扱いが難しいという問題がある。近年では計算機の性能の飛躍的な向上と、八分木を用いた効率的なアルゴリズムなどにより、ボクセルデータによる形状表現も一般的に扱われるようになってきている。

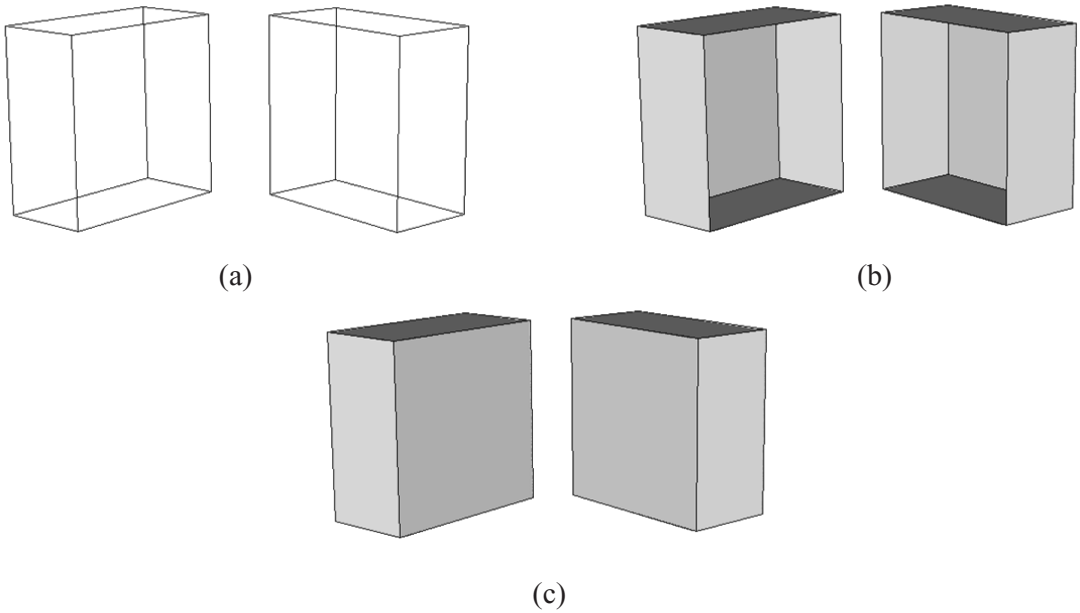


図 3.1: 3 次元形状モデルの表現方法
(a) ワイヤーフレームモデル (b) サーフェスモデル (c) ソリッドモデル

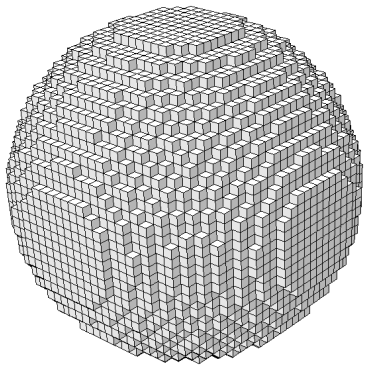


図 3.2: ボクセル表現による球体

CG の世界では、計算機内に構築された仮想的な 3 次元形状の外観さえ現実的に見えれば十分なことが多く、主にサーフェスモデルで定義される形状表現を用いることが多い。一方で CAD の世界では、現実の世界で作成できる「モノ」の設計が前提にあるため、厳密に立体の内部も定義されているソリッドモデルを用いることが多い。

ところで、一般的な紙模型は、立体の内部は考慮せずに「表面」だけを紙で表現することで、低コストで手軽に形を作成できるという特徴がある。計算機内で構築された 3 次元形状が、

サーフェスモデルとして表現されていれば、そのサーフェス（面）を平面に展開することで、立体模型を作成することができる。またソリッドモデルについても、その中と外を定義する境界によって表現された境界表現モデルについては、これと同様に境界面を平面に展開することで立体模型を作成できる。一般の CG や CAD では、サーフェスモデルや境界表現モデルが用いられることがほとんどであるため、計算機内に存在する既存の 3 次元形状データを紙模型にすることは、本質的に難しいことではない。しかしながら、円錐面や円柱面などの可展面以外の曲面は、歪み無く平面に展開することは不可能であるため（2.2節参照）、任意の自由曲面を含む 3 次元形状を紙模型で厳密に作成することは不可能である。そこで、本研究では、主にポリゴンモデルと呼ばれる、多角形の集合で表現される形状を扱うものとする。自由曲面をポリゴンモデルによって近似表現することは、 CG の世界でもよく行われている。ポリゴンモデルについての詳細は次節で述べる。

3.1.2 ポリゴンモデル

自由曲面形状の近似の一手法として、曲面上に配置された点群を直線の稜線で接続し、曲面を平面多角形によって近似する手法がある（まれに平面に乗らない多角形も許容される場合もある）。この多角形の集合で表現された形状を**多面体モデル**と呼ぶ。また、構成要素の面を**ポリゴン**、多面体モデルのことを**ポリゴンモデル**と呼ぶこともある。 CG の世界では一般的に後者の方がよく使われているようであるため、本論文ではこれ以降、「多面体モデル」を「ポリゴンモデル」と表記することとする。特にこの中で小さな多数の面の集合で表現したものを**多角形メッシュモデル**と呼ぶこともあるが、多面体モデルとの違いが明確に定義されているわけではない。図 3.3は球や立方体などの幾何基本形状（プリミティブ形状）のポリゴンモデルである。

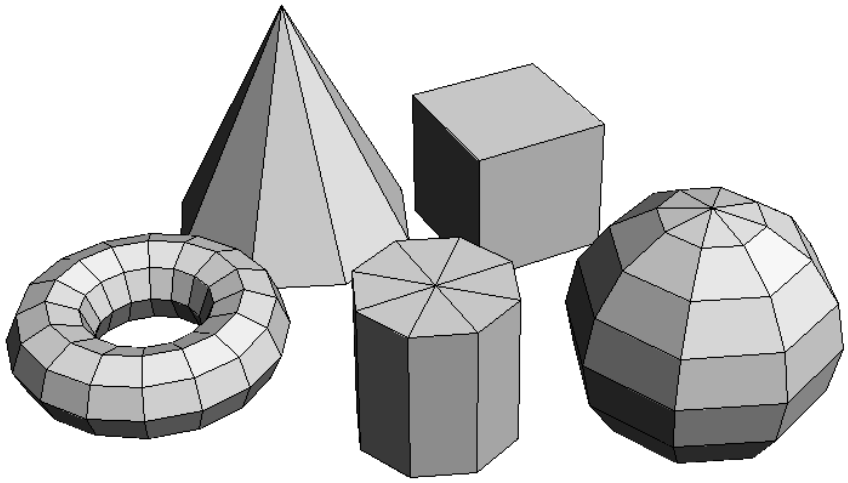


図 3.3: プリミティブ形状のポリゴンモデル

多面体モデルの幾何形状は、面、エッジ、頂点の 3 種類の要素と、頂点の幾何情報を表す 3 次元空間上の座標値、及びそれらの要素間の接続情報によって構成され、そのデータ構造には**ハーフエッジ構造** [33] が用いられることが多い。

図 3.4にハーフエッジデータ構造による頂点、エッジ、ハーフエッジ、面の各要素に関する表現を示す。頂点 (Vertex) は 3 次元空間上の座標値 (coordinate) を持つ。エッジ (Edge) はその両端の頂点 (start-vertex, end-vertex) を情報として持つ。ハーフエッジ (HalfEdge) は始めの頂点 (vertex) と、その頂点の周りに半時計方向に回った時の次のハーフエッジ (next)、対のもう一方のハーフエッジ (mate) を情報として持つ。面 (Face) は始めのハーフエッジ (start-halfedge) を情報として持つだけでよい。このようなハーフエッジデータ構造には、各要素から別の要素の探索を短い時間で容易に行えるという特徴がある。例えば、ある面 F に隣接する面を全て探索するには、 F に属するハーフエッジ he について $he \rightarrow mate \rightarrow face$ を調べればよく、全ての面を調べる必要がない。

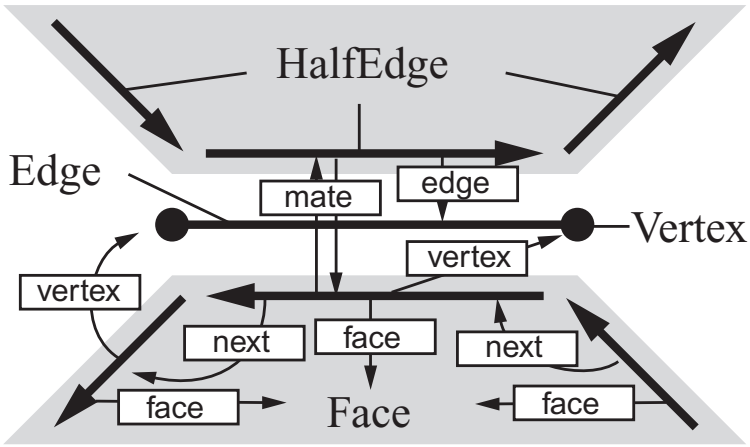


図 3.4: ハーフエッジ

また、多面体モデルのうち、特に小さな三角形のみの集合で表現したものが**三角形メッシュ**である。三角形は必ずただ 1 つの平面に乗ることが自明であるため、面の法線が厳密に求まり、またデータ構造をシンプルにできることなどから、計算機でデータを扱う際のメリットが大きい。一般的な多面体モデルも、各面を三角形に分割することで三角形メッシュとして表現することができる。

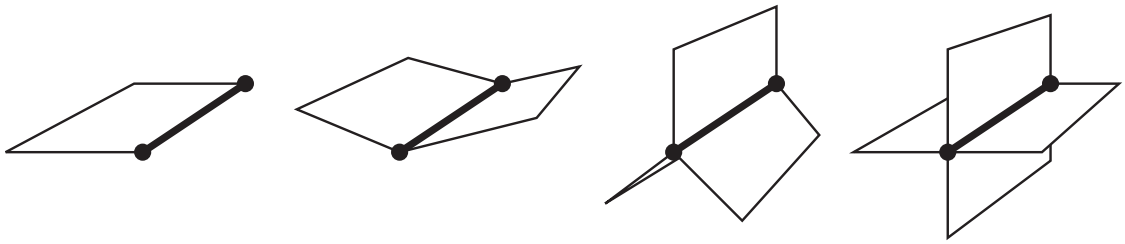
ところで、紙模型で形を作る場合には立体形状を展開図にする必要があるが、 2.2節で述べたようにどのような曲面も平面に展開できるわけではない。しかし、ポリゴンモデルは平面多角形の集合であるため、（複数のパーツに分かれるかもしれないが）どのようなものでも必ず平面に展開できるという特徴がある。このような観点から、 CG や CAD によってポリゴンモデルとして表現された立体形状は、紙模型のための形状表現に適していると言える。

3.2 ポリゴンモデルの展開

ポリゴンモデルは構成要素が平面多角形であるため、その展開図の作成は曲面に比べると単純である。しかし、計算機で表現されたモデルから展開図を作成するにあたっては、そのデータの扱い方とアルゴリズムを定義する必要がある。

本節では、計算機で表現されたポリゴンモデルについて、その展開図の作成方法の基礎的な内容と、それを計算機で行う際に必要となるアルゴリズムについて述べる。

なお、ここで扱うポリゴンモデルは 3.1.2節で述べたハーフエッジ構造によって表現されるものとする。ハーフエッジ構造の性質から、図 3.5(c)、(d) のような三面稜線（3 つの面に属するエッジ）や、4 つ以上の面に属するエッジは含まれず、(a)、(b) のようにエッジは必ず 1 つまたは 2 つの面に属す。



(a) (b) (c) (d)

図 3.5: 面とエッジの関係

3.2.1 位相的な視点から見た展開

ポリゴンモデル内で隣接する 2 つの面の間には、その 2 面を介するエッジが 1 つだけ存在するため、ポリゴンモデルの面を節点、エッジを枝とした、面の接続を表すグラフを作成できる。これを**面グラフ**と呼ぶこととする。図 3.6(a) に示す 6 面体の面グラフは図 3.6(b) のように表すことができる。節点から出る枝の数は、その面のエッジの数に等しく、例えば 3 角形メッシュモデルの場合、節点には 3 本の枝が接続することになる。

このポリゴンモデルの展開図（図 3.6(c)）の面グラフは、すべての面が連結しており、かつ閉路を持たないため、接続関係は図 3.6(d) の太線の様になり、展開前の面グラフの**極大木**となっていることがわかる。つまり、ポリゴンモデルの展開図の生成は位相的には連結な面グラフの極大木を求めることに等しい。一般にグラフ構造の極大木は、構成する枝の選び方によって膨大な数の組み合わせが存在し、この極大木の数だけ展開図の生成パターンが存在することになる。

ただし、幾何学的な視点から展開図の作成を考慮した場合、面と面が展開平面で干渉しないようにする必要があるため、すべての極大木のパターンから妥当な展開図を作れるわけではない。場合によっては、面と面の干渉を起こさずに展開できる極大木が 1 つも存在しないこともあり、この場合は展開図を複数のパーツに分ける必要がある。これは、面グラフを複数の連結グラフに分割することに等しい。

展開後の面グラフに存在する枝に対応する辺は、展開後も面と面とを接続する辺であり、展開図では山折りまたは谷折りの辺となる。一方、展開後の面グラフには存在しなくなった枝に

対応する辺は、展開する時に切断される辺であり、展開図の中で分離された 2 つの面の輪郭線の一部となり、展開図では切断線となる。

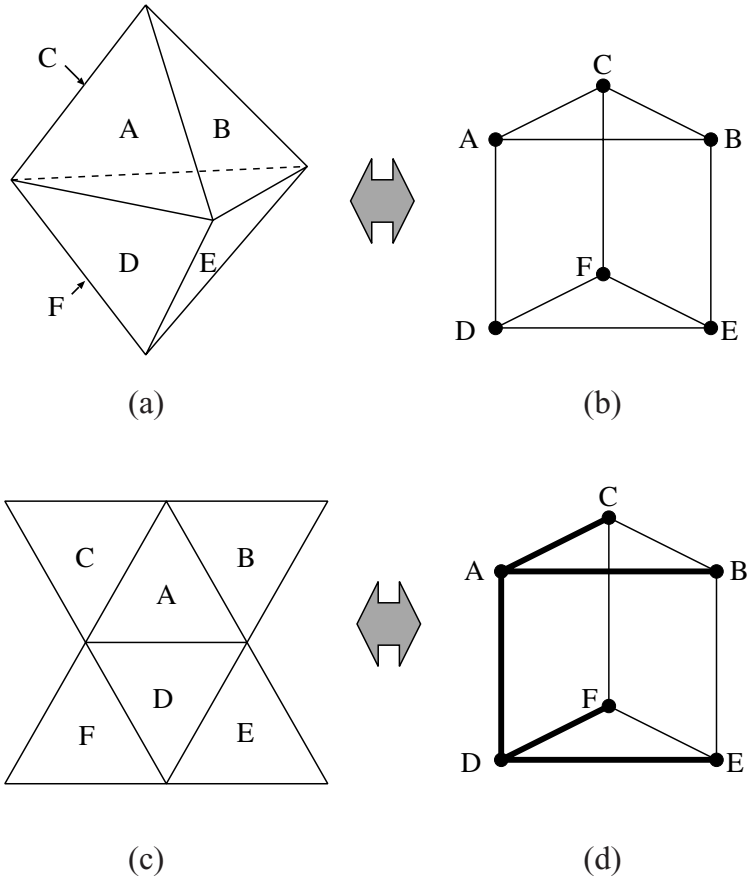


図 3.6: ポリゴンモデルの展開とそれに対応する面グラフ

3.2.2 幾何学的な視点から見た展開

ポリゴンモデルの展開図は、ポリゴンモデルを構成する各多角形を形を変えずに平面へ写像したものである。形の変形を伴わない射影変換を**剛体変換**と呼び、この変換による点 P の移動後の点 P' は、回転を表す直行行列$\boldsymbol{R}$と、平行移動ベクトル$\boldsymbol{T}$を用いて、$P' = \boldsymbol{R}P + \boldsymbol{T}$ と表すことができる（回転と行列を同時に表すことができる 4×4 の**同次行列**$\boldsymbol{M}$を使用すれば、$P' = \boldsymbol{M}P$ と表せる）。前節で述べた、展開図の面接続グラフが求まっているのであれば、それを元に隣接する面との位置関係から各面を剛体変換する$\boldsymbol{R}$と$\boldsymbol{T}$を順次求め、これによって**展開平面**（展開図を配置する平面）上に写像すればよい。図 3.7は、展開平面を xy 平面に定め、ポリゴンモデルの一部の 2 つの面を展開している様子を示す。三角形 $T_0(P_0P_1P_3)$ は $T'_0(P'_0P'_1P'_3)$ へ、$T_1(P_1P_2P_3)$ は $T'_1(P'_1P'_2P'_3)$ へ剛体変換によって投影される。T_0 を最初に展開される面とする場合は、T_0 は展開平面の任意の場所に写像できるが、次に展開される T_1 は T_0 との隣接関係を保持する場所へ写像する必要がある。この例では、3 次元空間で 2 面に共有される辺 P_1P_3 が、展開平面でも共有されるような剛体変換を行わなければならない。

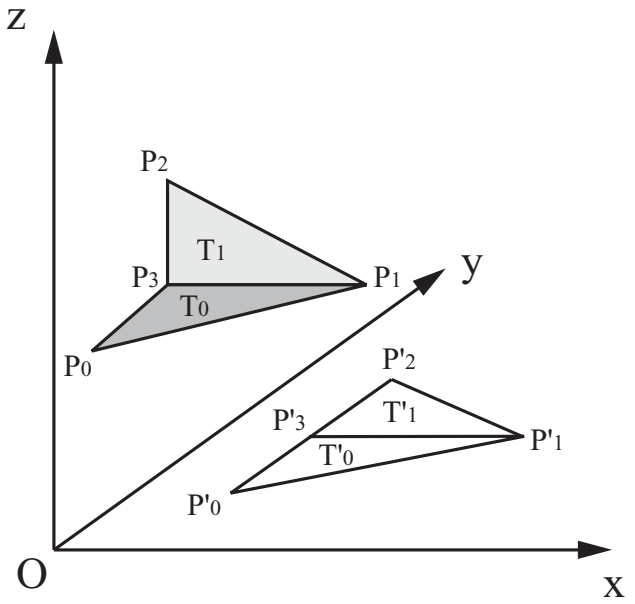


図 3.7: 剛体変換による 3 次元面の展開平面への写像

3.2.3 展開図作成アルゴリズム

これまでに述べたように、展開図の面グラフを定めてから、それを元に各面の展開平面上の座標を求めることで、ポリゴンモデルの展開図を生成できる。しかし、実際には展開平面上の各面の座標を求めるときに、面同士が干渉しあい、紙模型には適さない展開図が生成されることがある。このように展開図の面グラフを先に作成してしまう方法は、後から実際に展開してみるまで面の幾何的な干渉を判定することができず、干渉が見つかった場合には別パーツへの分割を行うか、もしくはそれを避けるためには面グラフの決定を最初からやり直さなければならないため、非常に効率が悪い。

これとは別に、最初に展開する面を 1 つだけ定めた後、干渉チェックを行いながら次に展開する面を順に決定してゆくと、干渉を常に回避しながら展開図の作成を進められるので効率がよい。この方法は言い換えると、面グラフの極大木を 1 枝ずつ成長させることと、展開図上の座標値を算出することを同時に進めてゆく方法である。

この方法による展開図作成は図 3.8に示すアルゴリズムで実現できる。まず始めに、最初に展開する面を決定し、それを展開平面上に配置する。次に、まだ展開されておらず、展開済みの面に隣接する面を次に展開する「候補」とする。ここで、候補としたのは、実際には展開平面上で他の面と干渉して展開できない可能性があるからである。なお、他の面と干渉することが既にわかっている場合は、この面と既存の展開済みの面の間にあるエッジにフラグ（図 3.8中の「NG フラグ」）を立てておき、同じ判定を繰り返し行わないようにしておく。選択した面を平面上に配置しても既存の展開済みの面と干渉しないことが確認できた場合には、この時点で初めて展開を行う。この後は、同じように次に展開する面の候補を探し、同じ処理を繰り返し行う。次に展開可能な面の候補が無くなった場合、全ての面が展開されていれば処理は終了する。まだ未展開の面が残っている場合には、別のパーツとして、改めて最初の面の決定から処理を開始する。

なお、展開を行う時に必要になる、多角形の展開平面上での座標の求め方と、干渉判定の方法については次節以降で詳しく述べる。

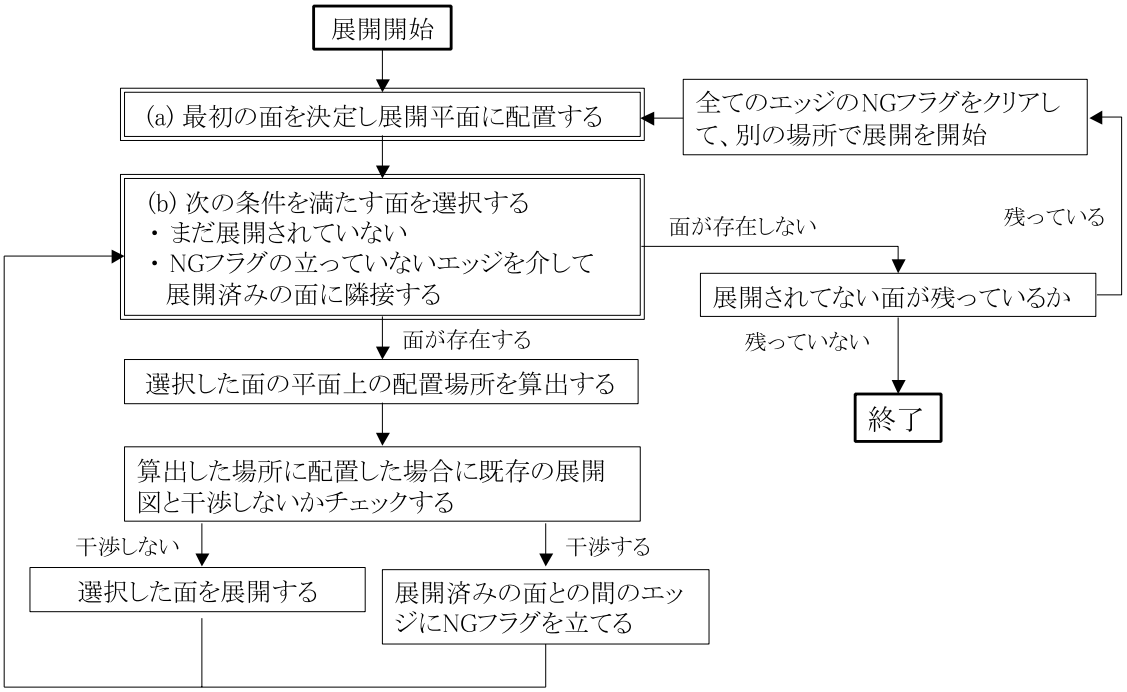


図 3.8: ポリゴンモデルの展開の流れ

ところで、図 3.8では、図中の二重枠で囲まれた (a) の「最初の面の決定」と (b) の「次に展開する面の決定」のステップにおいて、ほとんどの場合に多数の選択候補が存在する。この候補の面のうち、どの面を選択するかによって、最終的に生成される展開図が異なるものになる。この面の選択方法についての考察は 3.4節で詳しく述べる。

3.2.4 展開後の 2 次元座標の算出

3.2.2節では、展開図を生成するには各面ごとに回転行列$\boldsymbol{R}$と平行移動ベクトル$\boldsymbol{T}$を求めて、3 次元空間から展開平面への剛体変換を行えばよいと述べた。しかし、展開平面上での各頂点の座標値を求める問題は、このような剛体変換のための行列を求めずとも、三角形の三辺相同の性質（「三角形の 3 辺の長さが既知であれば三角形の形は一意に定まる」という性質）を用いることで、簡単な幾何の問題に置き換えることができる。以降、この手法について説明する。

前節で述べたアルゴリズムの流れで展開図を作成する場合、次に展開する候補となる面は、既に展開済みの面と必ず 1 辺を共有して隣接している。既に展開されている面の頂点は、展開平面での座標値が既知であるから、例えば次に展開する面が 3 角形である場合には、展開済みの面と共有する辺に含まれない残りの 1 点の座標値だけ求まればよいことになる。図 3.9の例のように、次に展開する面が三角形 P_0P_1Q で、既に展開済みの面と辺 P_0P_1 を共有する場合、残りの 1 点 Q の展開平面での座標がわかればよい。三角形 P_0P_1Q の各辺の長さを l_0, l_1, l_2 とすると、展開平面上の図 3.9において、点 Q の座標は点 P_0 を中心とする半径 l_2 の円と、点 P_1 を中心とする半径 l_1 の円の交点として容易に求めることができる。交点は、図 3.9の点 Q, Q'

のように 2 箇所求まるが、多角形を構成する頂点列の巡回方向が裏と表でどちら向きになるかは既知であるため、一意に決定することができる。面の表向きが反時計回りの頂点列によって定義される場合、図 3.9の例では点 Q が選択される。特に面が三角形からのみ構成される場合、この方法で高速に展開平面上の点を求めることができる。

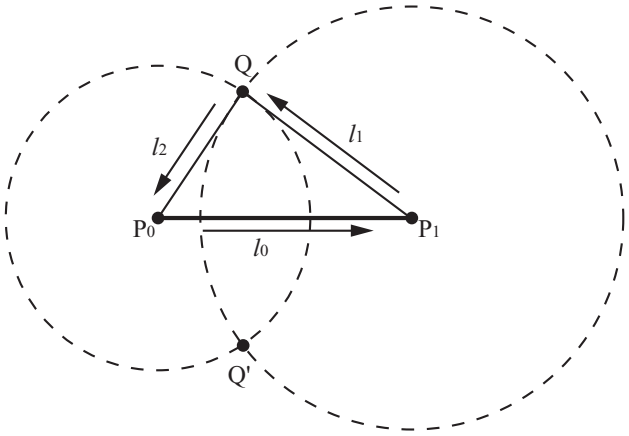


図 3.9: 3 辺の長さを用いた平面上の座標決定

上で述べた方法は、面が三角形である場合だけを考慮していたが、面が 4 角形以上である場合にも、図 3.10のように各面を三角形に分割することで、全ての面が三角形で構成されているものと同じように扱うことができる。ところで、 OpenGL などの三角形分割ライブラリを用いると、図 3.11のように、隣接する三角形が共通する辺を持たないような分割結果が得られることがある。この場合、ここで提案したような、以前に展開された面と共有する辺の座標値を用いた展開ができないという問題が生ずる。

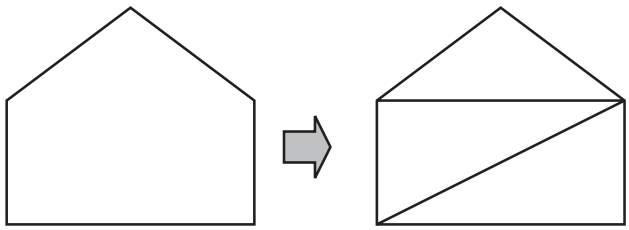


図 3.10: 多角形の三角形分割 (1)

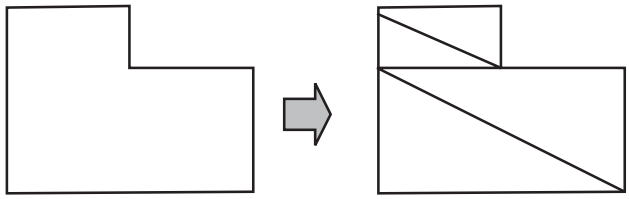


図 3.11: 多角形の三角形分割 (2)

そこで、三角形分割を用いずに、次のようにして多角形の各頂点の座標を求める方法を考案した。

多角形が n 角形であり、それぞれの頂点を反時計回りに $P_i(1 \leq i \leq n)$ とするとき、辺 P_0P_1 を共有する面が展開済みであれば展開図上の P_0、 P_1 の座標値は既知であるから、 $i(2, 3, \ldots, n)$ 番目の座標値は P_iP_{i-1}、 P_iP_{i-2}、 $P_{i-1}P_{i-2}$ を三辺とする三角形から求めることができる（図 3.12）。

しかし、これがうまく機能するのは多角形が凸であるときであり、非凸多角形である場合には、図 3.13の (d) のように、正しくない座標値が求まってしまう。これは、三角形を構成する頂点の巡回の向きが常に反時計まわりとみなしているためである。これを回避するために、 i 番目の頂点の座標を求める際に $i - 1$ 番目の頂点の角度を調べ、この角が 180 度より大きいときには、頂点の巡回の向きを逆にして求める必要がある（図 3.13(e)）。

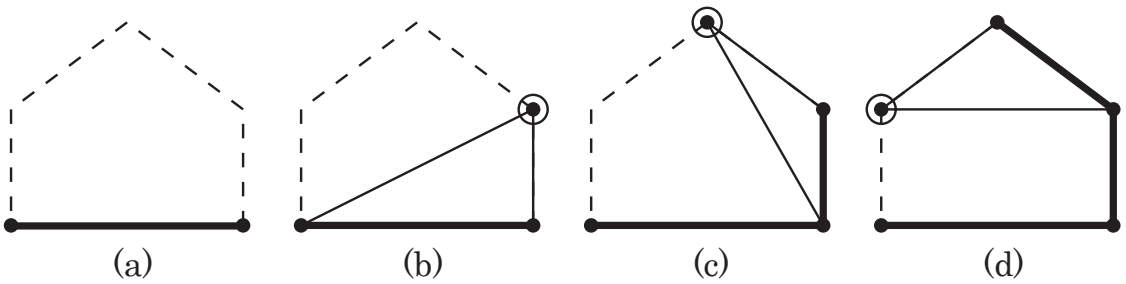


図 3.12: 多角形の平面上の座標決定

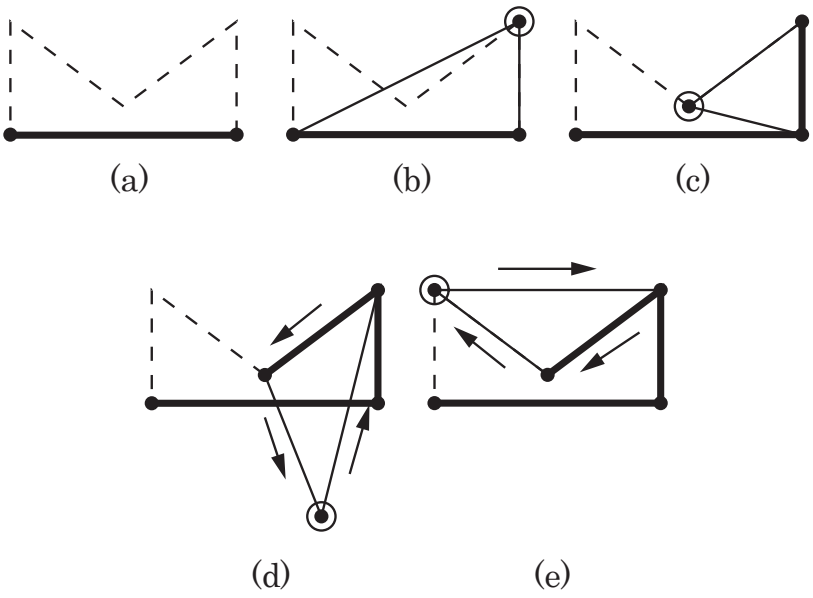


図 3.13: 非凸多角形の平面上の座標決定

3.2.5 展開時の面の干渉判定

展開図の作成は、前節で述べた方法で 1 つ 1 つの面を順に展開図平面へ配置してゆくことで行える。その際に、新たに展開図平面へ配置される面が、既に展開図平面へ配置済みの面と干渉しないようにチェックする必要がある。もし、新しく配置される面が既存の面と干渉するようであれば、別の面から展開を進めたり、新しいパーツとして新規に展開図作成を開始するなどの回避が必要となる（3.2.3節参照）。

新規の面と既存の面の干渉判定は、それぞれの面によって定義される領域の論理積（2 つの領域の交わってできる新しい領域）がゼロであるか否かによって明確に判定できるが、計算機上での論理演算の実装とその処理には一般に大きなコストがかかる。特に演算誤差を考慮すると、乾 [34] がまとめているように厳密な判定は難しい。

ここでは、単純な線分の交差判定と特定の点の内包チェックのみで面の干渉を判定する手法を提案する。

線分の公差判定を用いた干渉チェック

2 つの異なる面の最も単純な干渉判定は、それぞれの面を構成する線分同士が交差しているか否かを調べることで行える。例えば図 3.14では、辺 P_2P_3 と P_4P_7、 P_4P_5 と P_1P_2 が交わっているため、面 F_0 と F_1 は干渉する領域を持つことを容易に判定できる。

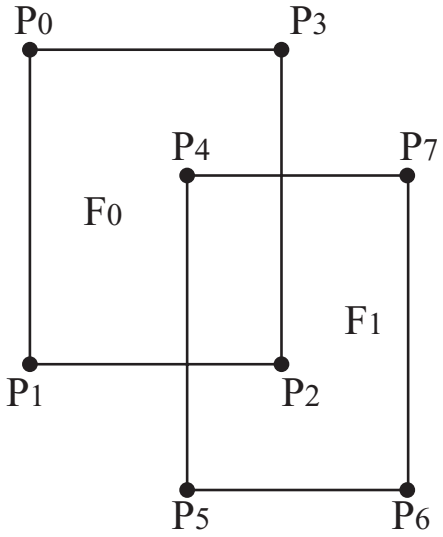


図 3.14: 線分の交差判定による面の干渉判定

ところで、図 3.15(a) のように二つの線分が重なっていたり、 (b)、 (c) のように一方の線分の端点が他方の線分の上に乗っている場合、これを線分の交差と判定するか否かが問題となる。

仮に、図 3.15の状態を「交差している」とみなした場合、図 3.16(a) のように、本来 1 つのつながった面として展開されることが望ましい角柱の底面部が、干渉判定を回避するために、一続きの連続した面として展開されない、という問題が起こる。工作の上では問題ないが、組

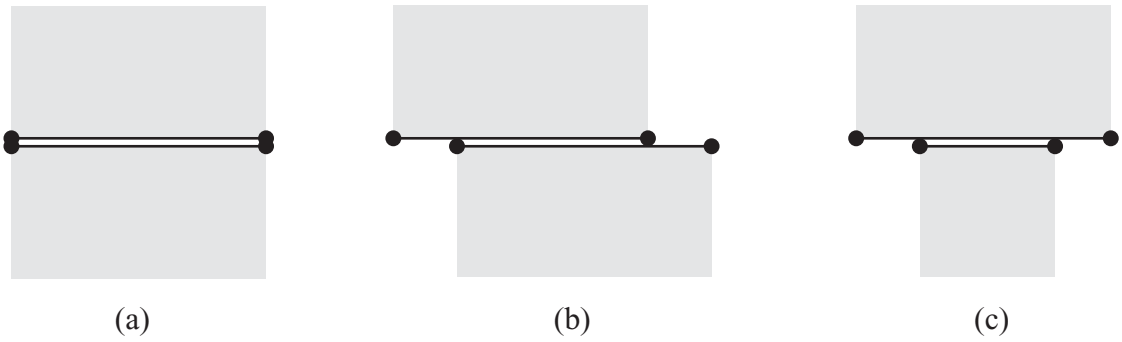


図 3.15: 線分の交差判定

み立てやすさを考慮すれば図 3.16(b) のような展開図を得られることが望ましいのは明らかである。

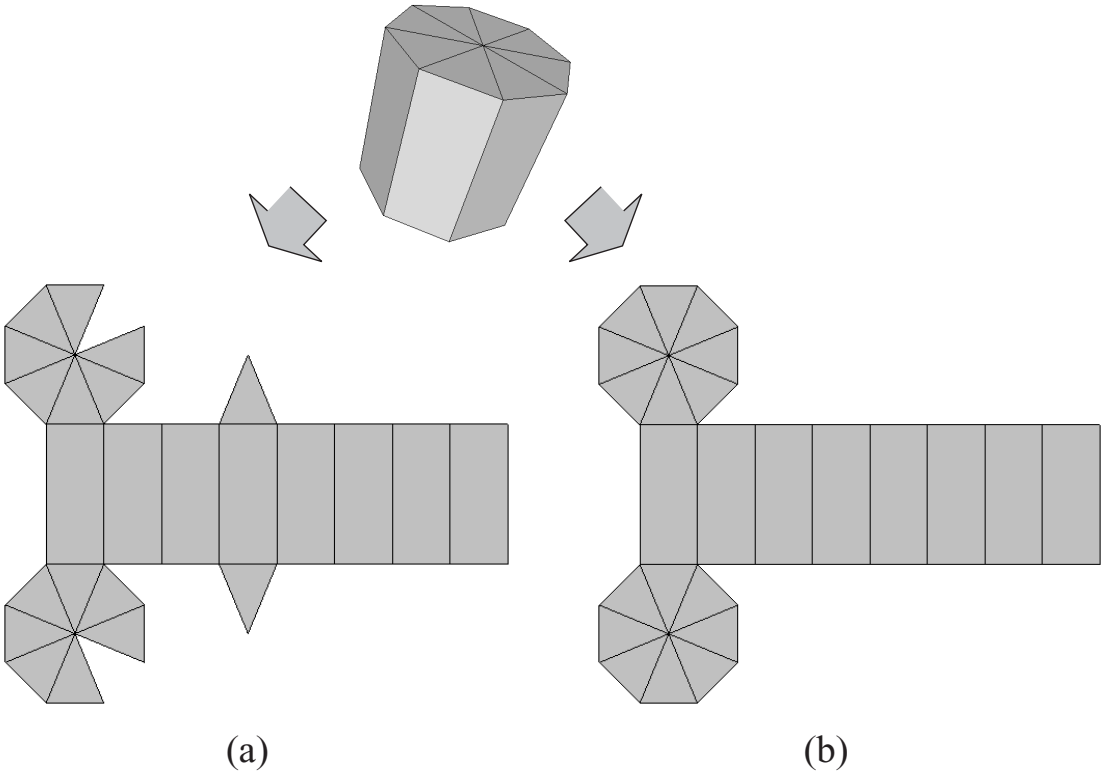


図 3.16: 底面が分離した角柱の展開図

前述の問題を解消するために、図 3.15の状態を「交差していない」と判別した場合はどうであろうか。今度は図 3.17のように、展開したときにまったく同一の場所に配置される面（図中斜線塗りの面）の重なりを判定できないという問題が起こる。

以上より、線分の交差判定だけでは、望ましい展開図を生成するための面の干渉を判別できない、と言える。そのため、線分の交差判定以外の方法で面の干渉を判別する手法が必要がある。

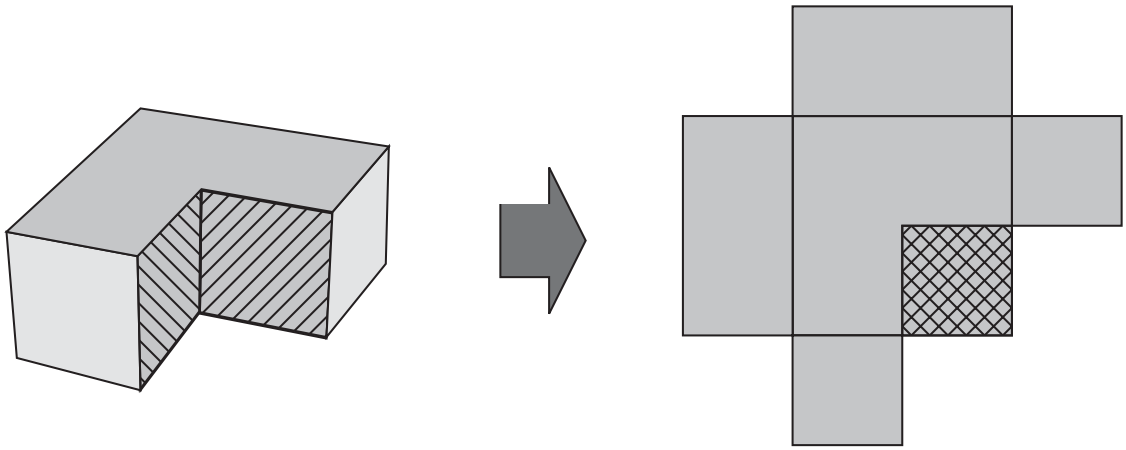


図 3.17: 展開図で面が重なる例

点の内包判定を用いた干渉チェック

ここでは、図 3.15の状態を「線分は交差していない」とし、交差する線分が存在しない場合にはさらに「一方の面に含まれる点が他方の面にも含まれるか否か」をチェックすることで、面の干渉を判定する手法を提案する。

干渉チェックを行う 2 つの面を A と B とした場合、この 2 つの輪郭線が互いに交差していないと判定された場合、A と B のありうる関係は $A = B$、$A \subset B$、$B \subset A$、または $A \cap B = \phi$ のいずれかである（$A \subset B$ は A が B の部分集合であることを示すものとする）。ここで、$A = B$、$A \subset B$、$B \subset A$ の場合は「干渉している」、$A \cap B = \phi$ の場合は「干渉していない」と判定したい。

ところで、A 上のある点 P_A が B に含まれる場合は $A = B$、$A \subset B$、$B \subset A$ のいずれかであり、必ず干渉している。一方、含まれない場合は $A \cap B = \phi$ または $B \subset A$ のどちらかであり、これだけでは干渉しているかどうかは判定できない。同様に、B 上のある点 P_B が A に含まれる場合は $A = B$、$A \subset B$、$B \subset A$ のいずれかであり、必ず干渉している。含まれない場合は $A \cap B = \phi$ または $A \subset B$ のどちらかであり、これだけでは干渉しているかどうかは判定できない。しかし、P_A が B に含まれないことがわかっている場合は、$A \subset B$ ということはありえないため、$A \cap B = \phi$ であることが確定し、この A と B は干渉していないことがわかる。

以上よりをまとめると、図 3.18に示す処理の流れで、面の干渉判定を正しく行えることがわかる。

なお、ここで述べる「一方の面に含まれる点」の具体的な例としては、一方の面を三角形分割した、ある三角形の重心を用いることなどが考えられる。面に含まれるかどうかの判定は、やはり面を三角形分割しておくことで、点と三角形の内包チェックという簡単な判定に置き換えることができる。

そもそも面同士の重なりは、面によって定義される領域の論理積がゼロであるか否かによって明確に判定できるが、一般に計算機上に厳密な論理演算を実装するのは難しい。本手法では、線分の交差判定と特定の点の内包チェックのみで面の干渉を判定できるため、実装とその処理のコストが小さく優れているといえる。

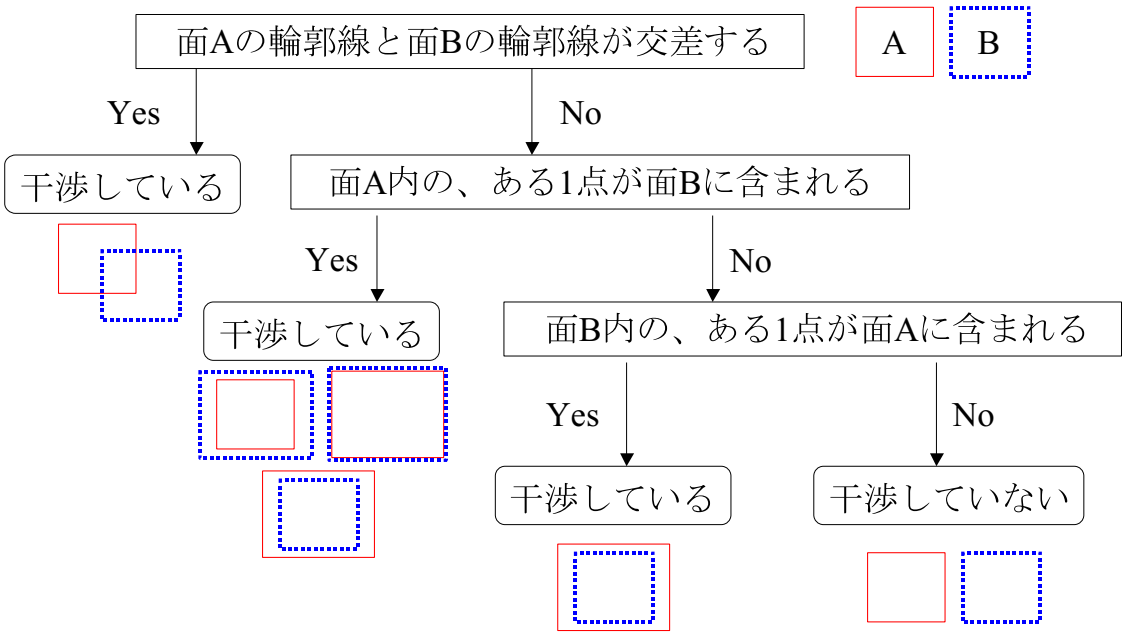


図 3.18: 干渉判定の流れ

3.2.6 ポリゴンモデルのデータ構造

ポリゴンモデルの表現には、一般的にハーフエッジ構造（3.1.2節）が用いられることが多い。ところで、展開図を作成するためには 3 次元の座標値だけではなく、展開後の 2 次元の座標値も保持しておく必要がある。一般に、展開前の頂点は、展開後には複数の頂点に分割されるため、この 2 次元の座標値はVertex （頂点）ではなく、Halfedge （ハーフエッジ）に保持しておくと都合がよい。また、 3 次元では隣接関係にあった面と面も、展開図では間にあるエッジが切断されて、隣接関係を失う場合がある。この情報は、展開図を構成する辺が切断線であるのか折れ線であるかの判定に必要であるため、やはりHalfedge に保持しておく必要がある。

また、詳細は 3.5節で述べるが、展開図に工作用の「のりしろ」を生成することや、テクスチャ画像を貼ることを考えると、さらに拡張が必要である。これらの情報はいずれもHalfedge に持たせることができる。このようにして、ペーパークラフト用の展開図のためにHalfedge を拡張したものをCraftHalfedge とすると、これは図 3.19のような擬似コードで定義できる。

```
// 一般的なハーフエッジクラス
class Halfedge {
public:
    Halfedge *prev;
    Halfedge *next;
    Halfedge *mate;
    Edge     *edge;
    Vertex   *vertex;
    Face     *face;
};

// 展開図生成用に拡張されたハーフエッジクラス (一般的なハーフエッジを継承する)
class CraftHalfedge : public Halfedge {
public:
    // 展開図を作成するために必要な拡張
    Boolean hasMateOnUnfoldedPattern;      // 展開図上で隣接する面があるか
    Point2d positionOnUnfoldedCoordinate; // 展開図上の頂点座標

    // その他の拡張
    Vector2d textureUV;     // テクスチャ用の UV 座標
    Boolean  hasFlap;       // 「のりしろ」をもっているか否かのフラグ
    double   flapWidth;     // 「のりしろ」の大きさ
    // etc...
};
```

図 3.19: ハーフエッジの拡張

3.3 展開図の組み立てにかかるコストの評価

多面体モデルの展開図作成手法についてを前節で述べたが、同じ多面体モデルを展開した場合にも、展開時に切り開かれる辺と接続される辺の決定方法により、結果として幾通りもの異なる展開図が生成される。この展開図の形状により、切断する長さや折り曲げの場所が異なるため、同じ立体を紙模型として作る時でも、工作しやすい場合と工作しにくい場合がある。しかし、工作の難易度を測定する手法は確立されていないため、どのような展開図がよいのかを客観的に判断するのは難しい。

ところで、曲面を含む形状においては、平面に歪みなく展開できるものは、その幾何的な性質により、錐面、柱面、接線曲面に限定されるため（2.2節参照）、これら以外の曲面を紙模型で表現するには、はじめに曲面を多面体モデルで近似し、それを展開することが行われる。多面体モデルによって曲面を精度よく近似しようとした場合、面の数は数千、数万のオーダーとなり、工作するには現実的でない展開図となることが多い。そこで実際に工作が行えるようにモデルを簡略化し、面の数を減らすことを行うが、その際にはどの程度まで面の数を減らすと、どの程度工作の手間が減少するのかを判断する必要がある。近似精度の維持と工作の手間の低減の間に生じるトレードオフの関係から最適な簡略化を求めるとした場合、工作の手間を数値で評価する手法が必要となる。本章では、展開図を切り抜いて紙模型を組み立てるのに要する**工作のコスト**を、与えられた展開図の幾何的な値から評価するための手法を提案する。

なお、組み立てにかかるコストを事前に評価することは、主に工業製品の設計・製造段階において重要であり [35]、図面から作りやすさを定量的に評価する研究などもなされている [36]。また、人間の手の動きを伴う作業に要する時間の見積もりや評価においてはヒューマン・インターフェースの分野で様々な研究がなされている [37]。ここでは、展開図から紙模型を作成する際にかかるコストの要因を、その工作の工程から 3 つに分類し、それらの評価式の作成と、簡単な実験によるパラメータの決定を行うこととする。

3.3.1 コストの分類

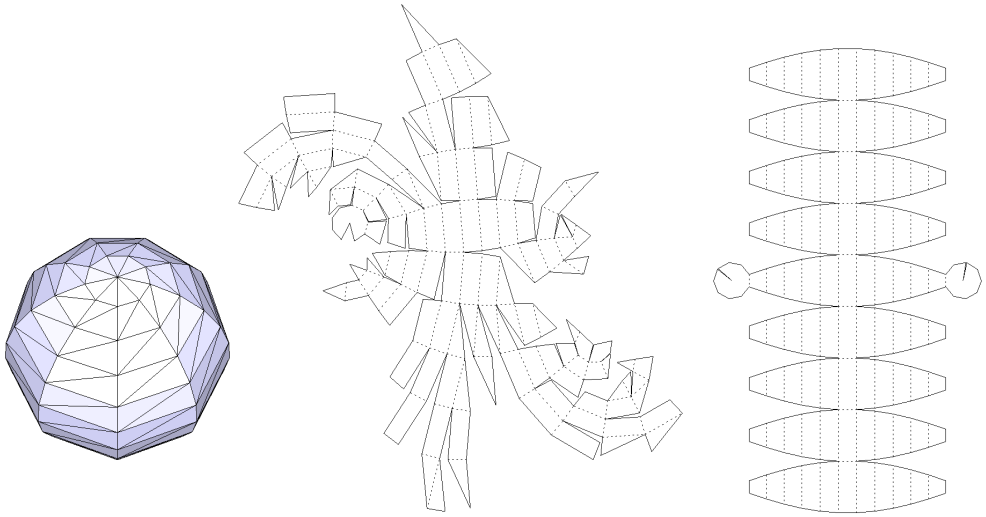
図 3.20の (a)、 (b) はどちらも同じ球体の展開図だが、一般には (b) の方がよい展開図と判断される。 (b) の方が「見た目が綺麗」という定性的な評価もあるだろうが、実際に工作して (b) の方が「工作にかかる時間が小さい」という定量的な評価も可能と思われる。ここでは後者の定量的な評価を用いて、展開図の良し悪しを判断するための基準を提案する。

具体的には、「展開図の組み立てにかかるコスト」＝「工作にかかる手間」とし、このコストを展開図の幾何的な値から評価を行うための「評価式」を提案する。

展開図から立体の紙模型を作成するのにかかるコストの大きな要因には、次の 3 つが挙げられる。

- 展開図の切り抜きにかかるコスト
- 展開図の折り曲げにかかるコスト
- 組み立て時の貼りあわせにかかるコスト

以降では、上記のそれぞれを「切り抜きコスト」「折り曲げコスト」「貼り合わせコスト」と呼ぶこととし、記号 $C_{cut}, C_{break}, C_{paste}$ と記すこととする。工作に要する総コストを C_{total} とすると、次の様な式で表現できる。



(a) (b)

図 3.20: 球の展開図

$$C_{total} = C_{cut} + C_{break} + C_{paste} \tag{3.1}$$

本論文では、これら 3 つの各コストを展開図の幾何的な性質から求める方法を提案する。ここでの「コスト」とは、単純に「作業に要する時間」に置き換えて考えることとする。また、工作の方法は用いる道具や貼り合わせ方などによって様々であるが、ここではカッターナイフで切り抜きを行い、また折り曲げ箇所はあらかじめカッターナイフで軽く半切り[1]を行い、貼り合わせにはセロハンテープを用いるものとして考える。

展開図の切り抜きにかかるコスト

切り抜きにかかるコストは、切断する辺の総延長（展開図の輪郭の総延長）に比例して増加すると考えられる。また、この切断辺の総延長が同じ場合であっても、実際に工作する際には図 3.21の (a) より (b) の方が切り抜きが大変である。これは、隣接する 2 つの切断線の向きが大きく変化する角が輪郭の中にある場合、そこで一度切り込みの向きと刃の初期位置の変更が必要であるため、その数に応じて切り抜きの中断に伴うコストがかかるものと考えられる。以降では、このような点を stoit 点[2]と呼ぶ。

切断する辺の総延長を L_{cut}、 stoit 点の数を N_{cut_stoit} とし、切り抜きコストに影響を与えるものがこの 2 つだけだと仮定すると、切り抜きにかかる総コストはそれぞれに重み係数 w_{cut_len}、w_{cut_stoit} を掛けて、次のように表せる。

$$C_{cut} = w_{cut\_lenLcut} + w_{cut\_stoit}N_{cut\_stoit} \tag{3.2}$$

ところで、直線部分を定規をあててカットする場合と、曲線部分をフリーハンドでカットす

[1]全部を切ってしまわずに、軽くカッターの刃でなぞる程度に筋を入れること
[2]stoit: つまずく

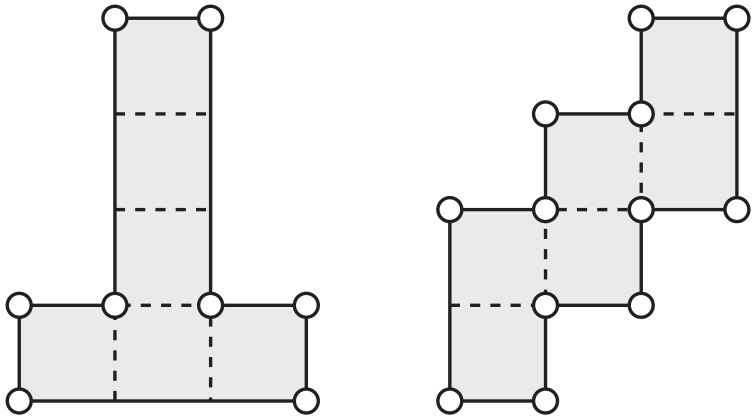


図 3.21: 展開図による stoit 点数の違い
stoit 点（図中の白丸）の数： (a) 8, (b) 12

る場合では、同じカット長でもコストが異なると考えられる。そこで、直線部分の長さを $L_{cut\_straight}$、曲線部分の長さを L_{cut_curve} とし、それぞれの重み係数を $w_{cut\_straight}, w_{cut\_curve}$ とすると、式 3.2は次のように表現しなおすことができる。

$$C_{cut} = w_{cut\_straight}L_{cut\_straight} + w_{cut\_curve}L_{cut\_curve} + w_{cut\_stoit}N_{cut\_stoit} \tag{3.3}$$

展開図の折り曲げにかかるコスト

折り曲げにかかるコストは、山折りまたは谷折りの辺の総延長および、折り曲げを要する辺の数に影響されると考えられる。折り曲げを行う辺はカッターで半切りをするとしているため、実際に折り曲げを行う作業に要するコストとともに、切り抜きのコストを考察した時と同様な半切りに要するコストと、 stoit 点と同様なカッターの移動と向きの変更に伴うコストも考慮すべきである。折り曲げの総延長を L_{break}、折り曲げる辺の数を N_{break} として、半切りに関する重み係数を $w_{halfcut}$、折り曲げに要するコストの重み係数を w_{break},、カッターの移動と向きの変更に関する重み係数を w_{break_stoit} とすると、折り曲げのコストは次のように表せる。

$$C_{break} = w_{halfcut}L_{break} + (w_{break} + w_{break\_stoit})N_{break} \tag{3.4}$$

組み立て時の貼りあわせにかかるコスト

貼り合わせにかかるコストは、今回はセロハンテープを用いるため、このセロハンテープを貼る箇所の数に比例すると考えられる。この数を N_{paste} とし、重み係数 w_{paste} 用いると、貼り合せのコストは次のように表せる。

$$C_{paste} = w_{paste}N_{paste} \tag{3.5}$$

3.3.2 展開図のコスト評価式

これまでに述べた、各要素を表現する式 3.3〜 3.5より、総合コストは次の式で表現できる。

$$\begin{aligned} C_{total} &= w_{cut\_straight}L_{cut\_straight} + w_{cut\_curve}L_{cut\_curve} + w_{cut\_stoit}N_{cut\_stoit} \\ &\quad + w_{halfcut}L_{break} + (w_{break} + w_{break\_stoit})N_{break} \\ &\quad + w_{paste}N_{paste} \end{aligned} \tag{3.6}$$

式 3.6に含まれる各重み係数の意味する内容は、表 3.1のようにまとめられる。なお、この式は展開図から得られる幾何的な値のみを考慮しており、「細かすぎると余計な手間がかかる」、「指が入りにくい形は作りにくい」などの、形状の大局的な特徴に依存したコストについては考慮していない。

表 3.1: コスト評価式に含まれる重み係数とその意味

パラメータ	内容
$w_{cut\_straight}$	定規にあてたカッターの刃が直線を単位長さだけカットするのに要する時間
w_{cut_curve}	フリーハンドでカッターの刃が曲線を単位長さだけ移動するのに要する時間
w_{cut_stoit}	定規にあてたカッターの刃が次のカットの準備に要する時間
$w_{halfcut}$	定規にあてたカッターの刃が直線を単位長さだけ半切りするのに要する時間
w_{break}	1 つの辺を折るのに要する時間
w_{break_stoit}	定規にあてたカッターの刃が次の半切りの準備に要する時間
w_{paste}	1 箇所貼り合わせるのに要する時間

3.3.3 重み係数の推定

前節でまとめたコスト評価式に含まれる係数について、その内容から

$$\begin{aligned} w_{halfcut} &= w_{cut\_straight} \\ w_{break\_stoit} &= w_{cut\_stoit} \end{aligned} \tag{3.7}$$

とみなし、 $w_{cut\_straight}, w_{cut\_curve}, w_{cut\_stoit}, w_{break}, w_{paste}$ の 5 つの値を実験から求めるものとする。

実験内容

実験には研究室の大学院生 5 人を被験者とし、前出の 5 つの係数を求めるための基礎実験（**実験 1**）とコスト評価式の妥当性を検証するための総合的な工作の実験（**実験 2**）を行った。

実験 1

図 3.22に示す例題を用い、 5 通りの作業についてそれぞれに要する時間を計測した。この計測内容とその意図は表 3.2の A 〜 E の通りである。なお、貼り合せに用いるセロハンテープは数センチの長さのものを、事前に必要な数だけ準備しておき、 1 辺に 1 箇所の貼り付けを行うものとした。

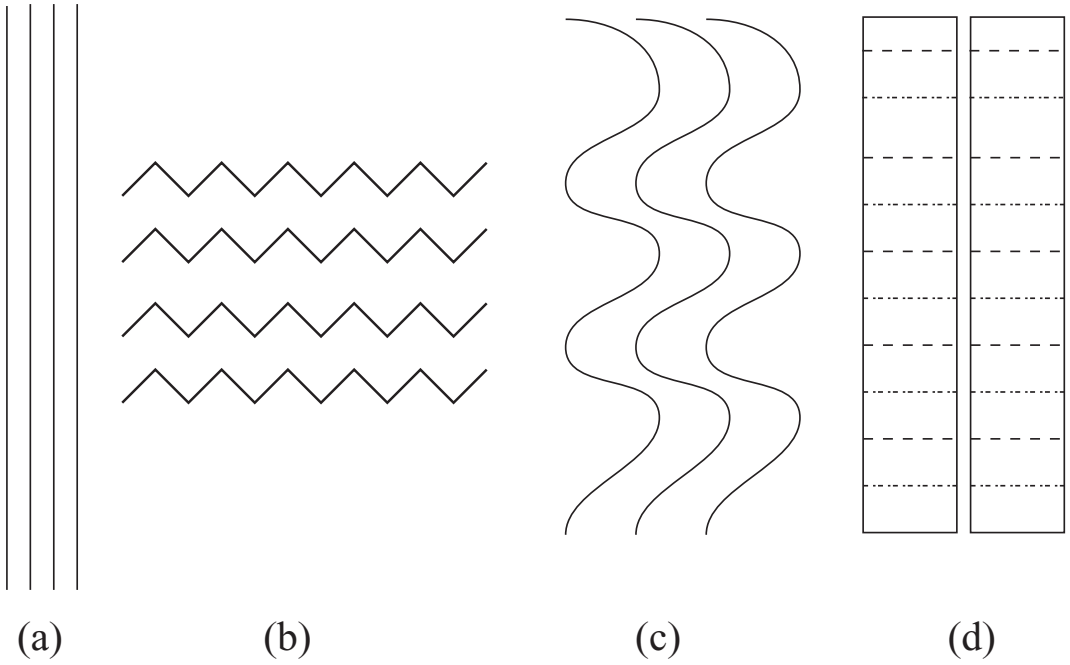


図 3.22: 実験 1 の例題
(a) 直線のカット (b) stoit 点を持つ直線のカット (c) 曲線のカット (d) 折り曲げ

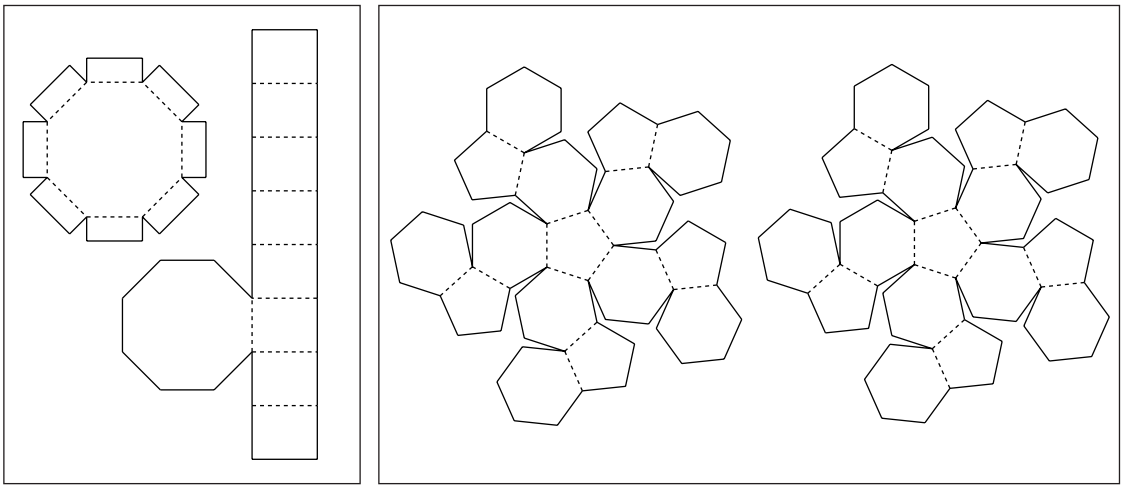
実験 2

図 3.23に示す例題を被験者に作成してもらった。図 3.23の展開図の (a) は蓋のある八角形の箱であり、 (b) はサッカーボールである。コスト評価式に含まれる (a)、 (b) の幾何的な値は表 3.3の通りである。なお、ここでの stoit 点は、輪郭上の両側の辺の成す角が 150 度以下の点であるものとした。また、工作の作業は実験 1 と同じ条件で行った。

実験結果

5 人の被験者に実験 1 の例題を与え、各課題について計測した平均値は表 3.4のようになった。これを元に式 3.6の各係数を算出した結果は、表 3.5のようになった。

実験 2 について、被験者が図 3.24のような紙模型の組み立てに要した時間の平均と、実験 1 によって導出された重み係数を用いて式 3.6より算出した値を比較した結果は表 3.6のようになった。



(a) (b)

図 3.23: 実験 2 の例題
(a) 八角形の箱 (b) サッカーボール

表 3.2: 実験の例題とその内容

例題	内容
A	図 (a) の長さ 25cm の直線を 4 回、定規を用いて切断し、それぞれの時間を合計する。（$w_{cut\_straight}$ を決定するため）
B	図 (b) の長さ 22cm、 stoit 点 10 の折れ線を 4 回、定規を用いて切断し、それぞれの時間を合計する。（w_{cut_stoit} を決定するため）
C	図 (c) の長さ 35cm 分の曲線を 3 回フリーハンドで切断し、それぞれの時間を合計する。（w_{cut_curve} を決定するため）
D	図 (d) の折る辺 10 箇所を折る作業を 2 回行い、それぞれの時間を合計する。（w_{break} を決定するため）
E	図 (b) と (d) を 11 箇所で貼り合せる作業を 2 回行い、それぞれの時間を合計する。（w_{paste} を決定するため）

表 3.3: 展開図の幾何的な値

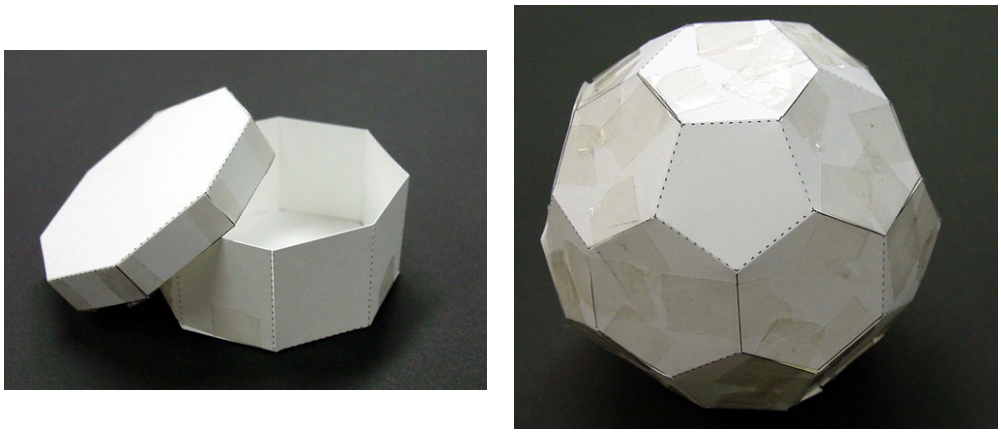
	$L_{cut\_straight}$	N_{cut_stoit}	L_{break}	N_{break}	N_{paste}
(a)	113cm	36	50cm	16	16
(b)	254cm	120	64cm	30	60

表 3.4: 例題の計測結果

例題	A	B	C	D	E
平均所要時間 (秒)	7.5	133	113	28	169
分散	4.8	234	18	46	861

表 3.5: 係数の算出結果

係数	値	単位
$w_{cut\_straight}$	0.08	sec/cm
w_{cut_curve}	1.1	sec/cm
w_{cut_stoit}	3.2	sec/stoit
w_{break}	1.5	sec/break
w_{paste}	7.7	sec/paste



(a) (b)

図 3.24: 実験 2 の例題の完成模型例
(a) 八角形の箱 (b) サッカーボール

表 3.6: 測定値と理論値

例題	平均測定値 (秒)	理論値 (秒)	誤差 (秒)	相対誤差
八角形の箱	359	327	32	0.09
サッカーボール	1140	1012	128	0.11

3.3.4 考察

実験 1 の結果より、定規を用いた展開図の切り抜きにおいては、直線のカット長は作業時間にほとんど影響を与えず、 stoit 点の数がより大きな影響を与えることがわかった。具体的には、 1 つの stoit 点は約 14cm の直線をカットすることに等しいコストであることが測定値より判断できる。 stoit 点においては定規の向きを変え、カッターを持ち上げるという作業が発生するため、作業コストが大きいと思われる。また、貼り合わせのコストについては、事前に適当な大きさに切り分けたセロハンテープを用い、表側から貼り付けるという、手軽な手法を用いたにもかかわらず、 stoit 点の 7 倍のコストがかかるなど、他の作業のコストに比べるとかなり大きく、工作の大きな割合を占めることがわかった。工作にかかるコストを小さくするためには、貼り合わせにかかるコストを小さくすることが重要であると言えそうである。

実験 2 を行った結果をまとめた表 3.6の、測定値と理論値の比較により、ここで提案した評価式は 10% 程度の誤差を含んで、実際に組み立てにかかるコストを予測できることがわかった。多くの仮定に基づいて作った評価式であったことを考えると、非常によい精度であると思われる。実際に測定した時間が理論値よりも若干大きい傾向にあるのは、評価式の中に含まれていない、工作の時のパーツの切り離しや工作用紙の取り替え、組み立て手順の検討時間などが影響していると考えられる。また、形状が複雑になるにつれ、思考に要する時間や集中力の低下などによるペースダウンなど、評価式で表せない要因も生じると考えられる。このため、より複雑な形では理論値と実際の値との差が拡大する可能性があるだろう。

今回の実験では、カッターと定規による切り抜きと、セロハンテープによる貼り合わせ方法を採用したが、実際にはハサミによる切り抜きや、ボンドによる貼り合わせを行うことも多く、また対象とする紙の厚さなどによっても、各種のパラメータは変化すると思われる。また、測定値にはばらつきが見られたため、作業の丁寧さや、個人の作業能力などに依存する点も大きいと思われる。今回は被験者が 5 人だけであったが、より多くの被験者を対象とした実験や、より多くの例題を用いた実験を行うことで、評価式に用いるパラメータの精度を高めることができると考えられる。ただし、作業の「慣れ」や「丁寧さ」によって、同じ被験者が同じ実験を行った場合でも、測定値に大きな違いが発生しがちであるため、これらの要素をどのように扱うべきかは難しい問題である。

また、例えばセロハンテープによる貼り合わせを「外側」からではなく「内側」から行うとした場合（貼り合わせ箇所を綺麗に見せるために、内側からセロテープで貼ることはよく行われる）は、「細い場所には指が入りにくくて手間がかかる」というような、展開図だけでは判断しにくい要因も含まれるため、これらを正確に評価することは難しいと思われる。

しかしながら、従来は実際に展開図を組み立てるまで工作の難易度がわからず、例えば面の数のみから判断するなどの方法しか存在しなかったため、工作の工程に基づいた評価式によって事前にある程度の難易度を判断できる本手法は、十分に意味のあるものだと思われる。また、 2 つの展開図があった場合に、どちらの方が組み立てやすいか、といった相対的な評価を行うには、今回の実験結果を用いることで妥当な判断ができるであろう。

3.4　ポリゴンモデルの展開法による組み立てコストの違い

3.2.3節で述べた展開図作成のアルゴリズムを用いてポリゴンモデルの展開図を作成する場合、図 3.8の二重枠で囲まれた (a) の「最初の面の決定」と (b) の「次に展開する面の決定」のステップでは、選択候補となる面が多数存在する。この選択候補の面のうち、どの面を選択するかによって、最終的に生成される展開図が異なるものになる。

特に「次に展開する面の決定」において、ある評価に基づいて次の面を選択することで、特徴のある展開図を得られることがある。例えば「そのときに展開図に存在する一番長い辺に接続する面を選択する」という工夫をすることで、全体的に折れ線となる辺の総延長が長くなり、結果として切断する辺の総延長が短い展開図が得られる。ここでは、この「次に展開する面の決定」の際に特定の評価基準を設け、その結果得られる展開図がどのようなものになるかを実験する。また、「最初の面の決定」が、どの程度得られる展開図のコストに影響を与えるかの実験も行う。

3.4.1　次に展開する面の決定方法と展開図のコスト

3.2.3節で述べたアルゴリズムを、図 3.8に記したフローに従って線形に処理を行う場合、常に次に展開可能な面の中から特定の面を選択して処理を進めることとなる。このように、2 つ先、3 つ先、といった複数手分の先読みを行わずに、1 手先の評価のみを考慮して、そのときに最適なものを選択する手法を**欲張り法**と呼ぶ。一方、フローを遡ることで、複数手分の評価を行うことも可能であるが、ポリゴンモデルの展開図の作成途中においては、たくさんの選択可能な面が存在し、このような複数手の先読みを含む探索を深めて行くと、すぐに組み合わせ数の爆発が起きてしまう問題がある。そのため、このような欲張り法によって近似的な最適解を求める方法は現実的な解法であるといえる。欲張り法で得た解は、特定の評価に基づく最適解を得られるわけではないが、簡単なアルゴリズムと高速な処理で、ある程度妥当な解を得られることが多い。

ここでは、3.3節で述べたコスト評価式を考慮して展開図全体の組み立てコストを小さくするために、図 3.8の (b) で示した「次に選択される面」を、以下の 3 通りの方法で決定し、欲張り法によって妥当な解を求めるものとした。なお、最初に展開する面は、最も長いエッジに隣接する面のどちらか一方とした。

1. ランダム　次の面の幾何的な性質に関わり無く、ランダムに面を選択する。これは、何の評価基準も設けずに展開図を作成した場合と、評価基準を設定して面の選択を行った場合とを比較するために行う。

2. 辺の長さ　そのときに展開図に存在する一番長い辺に接続する面を選択する。この方法は、折れ線となる辺の総延長を長くすることで、逆に切断すべき辺の総延長を短くすることを目的としている。

3. 辺の長さと角度

そのときに展開図に存在するなるべく長い辺に接続する面を選択するが、角度の小さい辺（隣接する面が平面に近い辺）が存在する場合にはこちらを優先する。この方法は、「平面に近い辺はなるべく切断されない方がよい」という人間の経験則に基づいて、辺

の長さを考慮しつつも、なるべく平面に近い辺は切断しないことを目的としている。具体的には両側の面の成す角が 170 度以上の辺を平面に近い辺とする。

上記の「ランダム」「辺の長さ」「辺の長さと角度」の各評価を、次の面を選択するときの評価基準に用い、それぞれ球（面数 216、高さ 10cm）と五角柱（面数 20、高さ 5cm）とトラの頭部のモデル（面数 246、高さ 10cm）の 3 つについて展開図を作成する実験を行った。ただし、「最初に展開する面」はランダムに決定するものとし、ここでは考慮しなかった。

結果として得られた展開図は図 3.25〜 3.27のようになった。それぞれの展開図の幾何的な値と、 3.3.1節で導出したコスト評価式（式 3.6式）と表 3.5にまとめた評価パラメータを用いた評価結果は表 3.7のようになった。

表 3.7: 展開図の幾何的な値と評価値

対象モデル	手法	$L_{cut\_straight}$	N_{cut_stoit}	L_{break}	N_{break}	N_{paste}	評価コスト
球	ランダム	449cm	175	239cm	135	109	2089
	辺の長さ	250cm	70	250cm	100	109	1573
	辺の長さと角度	250cm	70	250cm	100	109	1573
五角柱	ランダム	181cm	20	88cm	11	11	222
	辺の長さ	128cm	20	91cm	10	10	206
	辺の長さと角度	123cm	16	55cm	6	9	163
トラの頭部	ランダム	522cm	233	353cm	176	136	2690
	辺の長さ	401cm	195	383cm	172	130	2496
	辺の長さと角度	417cm	197	350cm	156	129	2418

3.4.2 最初に展開する面による展開図のコストの差異

前節では、「最初に展開する面」の選択を考慮せず、一番長い辺に隣接する面としていた。そこで、ここではこの「最初に展開する面」によって、展開図のコストにどの程度の差異が生ずるかを実験によって確かめる。

前節の「球体」や「五角柱」のような面の数が少ないモデルは、どの面からスタートしても得られる結果には差は無いと思われるため、ここでは前節で用いた「トラの頭部」のモデルを用いることとした。モデルに含まれる全ての面に 0 から 246 までの ID を割り当て、それぞれの面を「最初に展開する面」に設定した場合のコストを求めることで、最初に展開する面によってどの程度コストに違いがあるかを確かめた。なお、「次に展開する面」の決定方法には、前節で行った「ランダム」なものと「辺の長さと角度」の 2 通りについて行った。

実験の結果は表 3.8と図 3.28,3.29の通りである。表 3.8より、ランダムよりも、辺の長さと角度によって次に展開する面を定めた方が総じてコストが小さく、開始面に依存するコストのばらつきは小さいことがわかる。図 3.28,3.29からは、次に展開する面を考慮することで、開始面によるコストの違いが小さく抑えられていることがわかる。

また、前節で「辺の長さと角度」を考慮した場合に得られた結果の 2418 は、今回の実験の平均値に近いものであり、特にコストに特別な影響を与えるものではなかったことがわかる。

表 3.8: 最初に展開する面を変えた場合の展開図のコスト

次に展開する面の決定方法	平均	最大値	最小値	分散
ランダム	2642	2767	2500	1833
辺の長さと角度	2426	2453	2387	201

3.4.3 考察

「次に展開する面」の決定方法として、「ランダム」「辺の長さ」「辺の長さと角度」を考慮した 3 通りの方法を、単純な欲張り法を用いて実験した結果から、「辺の長さ」または「辺の長さと角度」を考慮した方がランダムな場合よりも全体的にコストが小さくなることを確認できた。この 2 つの方法については、特に「辺の長さと角度」を考慮した方が、平面に近い辺を切断辺とすることが少なくなるため、よりコストが小さくなることが確認できた。

また、このように「次に展開する面」を特定の規則に従って決定することで、球や五角柱のような、面の配置が規則正しい幾何形状については、展開図においても面の接続が規則正しくなるため、人間が見て組み立てやすいと感じる展開図を得ることができた。トラの頭部のモデルのように、規則的な幾何形状ではなく、面の数が多いモデルについては、展開図の見た目の違いはそれほど大きくないが、コストの値はやはり「辺の長さと角度」を考慮したものが一番小さくなることが確認された。このトラの頭部のモデルの例では、約 10% のコストの違いが生じた。

2 つ目の実験では、最初に展開する面を変えることで、その結果、コストがどのように変化するかを見ることができた。「次に展開する面」をランダムに決定した場合はばらつきが大きいが、「辺の長さと角度」を考慮した場合は、ほとんどばらつきが見られず、最大値と最小値でも 3% 程度の違いしかなかった。例えば異なる面から展開を開始しても、展開中で同じ形の展開図が現れた場合は、それ以降同じ面を選択してゆくことになるため、最終結果は同じものとなる。そのため、同一のコストとなるケースが多く、最終的なコストも小さな範囲に収まるものと思われる。ところで、この中で特にコストが最小となる面を、展開を始める前に予め見つけ出すことは難しい。全ての面について展開を試みることで、コストが最小となる展開図を得ることができるが、これは計算時間が面の数だけ増えることになり、わずかの違いのために、それほど時間を費やすメリットは少ないと思われる。

なお、ここでの実験では、欲張りアルゴリズムによって 1 手先のことだけを考えて次に展開する面を決定していたが、複数手先を先読みすることを行ったり、または遺伝的アルゴリズムや焼きなまし法などの、他の最適化のためのアプローチを用いることで、よりコストの小さな展開図を探し出すことも可能だと考えられる。ただし、最終的なコストは展開図が完成した後でないと算出できないため、膨大な展開図のパターンの中から最適解を求めることは非常に難しいと思われる。妥当な計算時間の中で、いかにコストの小さい展開図を得るかについては、まだ研究の余地が残されている。

また、今回は展開図のコストの評価に 3.3.2節で求めた展開図のコスト評価式を用いたが、この式では例えば展開図を構成するパーツの数などは考慮されていない。 1 つのつながったパーツに展開できる形状は問題ないが、例えばトラの頭部の展開図の例（図 3.27）のように、1 つのパーツには展開できず、複数のパーツに分かれてしまうような場合、パーツの数や、パーツの大きさ（パーツに含まれる面の数）のばらつきなども、評価の対象としたほうがよいかもしれないと感じられた。例えば面が 1 つしか存在しないようなパーツが多数生成された展開図

は、工作しにくい、工作の結果が美しくないなどの問題が生じると思われる。今回はパーツの分割についてはあまり考慮しなかったが、含まれる面が極端に少ないパーツは、他のパーツからいくつかの面を移動させて、なるべくパーツの大きさにばらつきをもたせないようにする、などの工夫も検討の余地があると思われる。

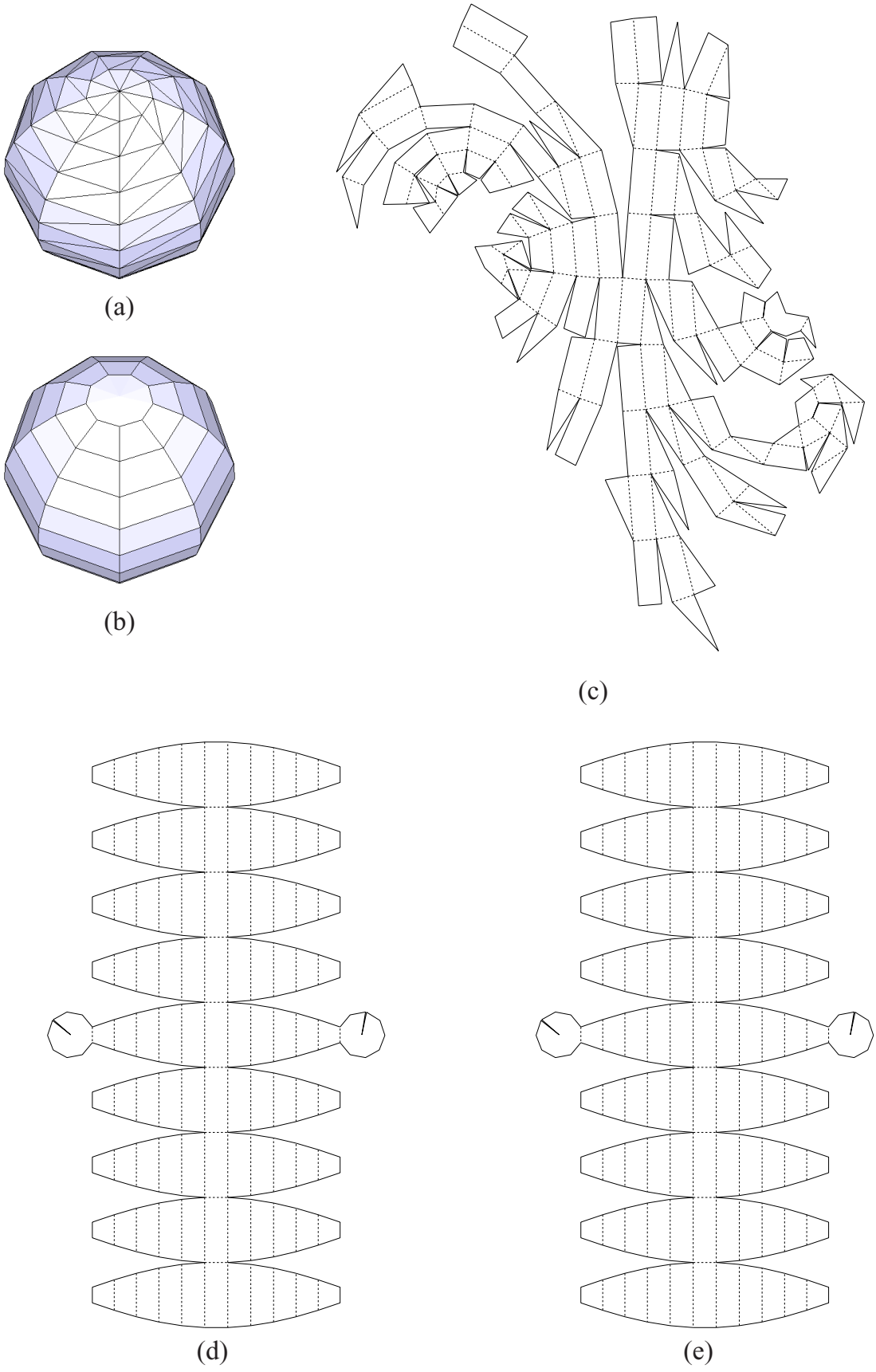
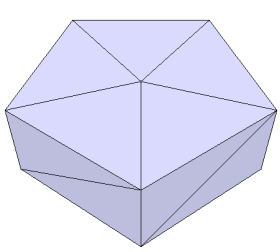
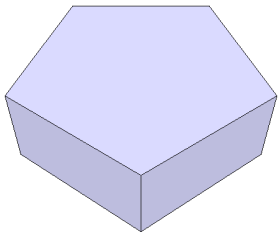


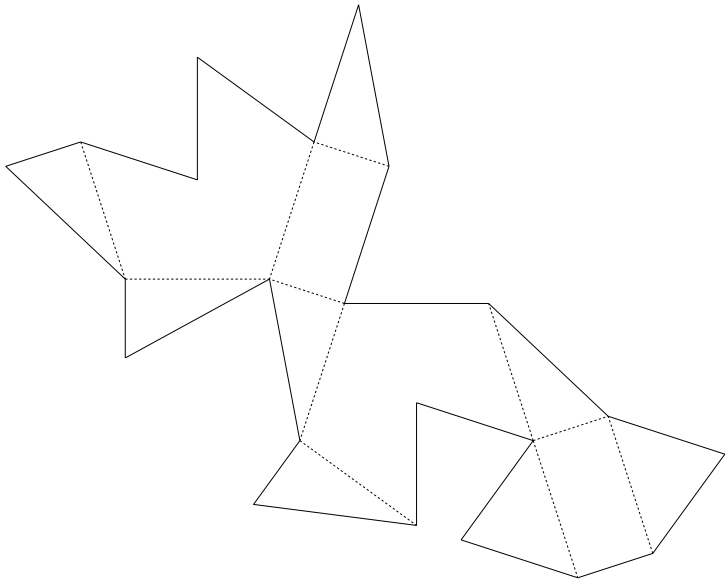
図 3.25: 球 (面数 216) の展開図
(a) 展開を行ったポリゴンモデル (b) 平面に近い辺を省略表示したポリゴンモデル
(c) ランダム作成した展開図
(d) 辺の長さを考慮して作成した展開図 (e) 辺の長さと角度を考慮して作成した展開図



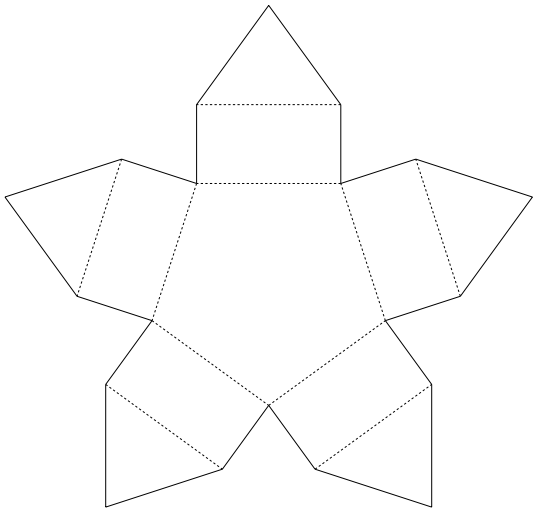
(a)



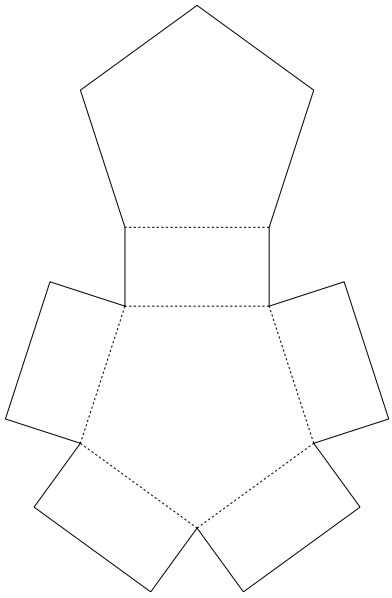
(b)



(c)



(d)



(e)

図 3.26: 五角柱 (面数 28) の展開図
(a) 展開を行ったポリゴンモデル (b) 平面に近い辺を省略表示したポリゴンモデル
(c) ランダム作成した展開図
(d) 辺の長さを考慮して作成した展開図 (e) 辺の長さと角度を考慮して作成した展開図

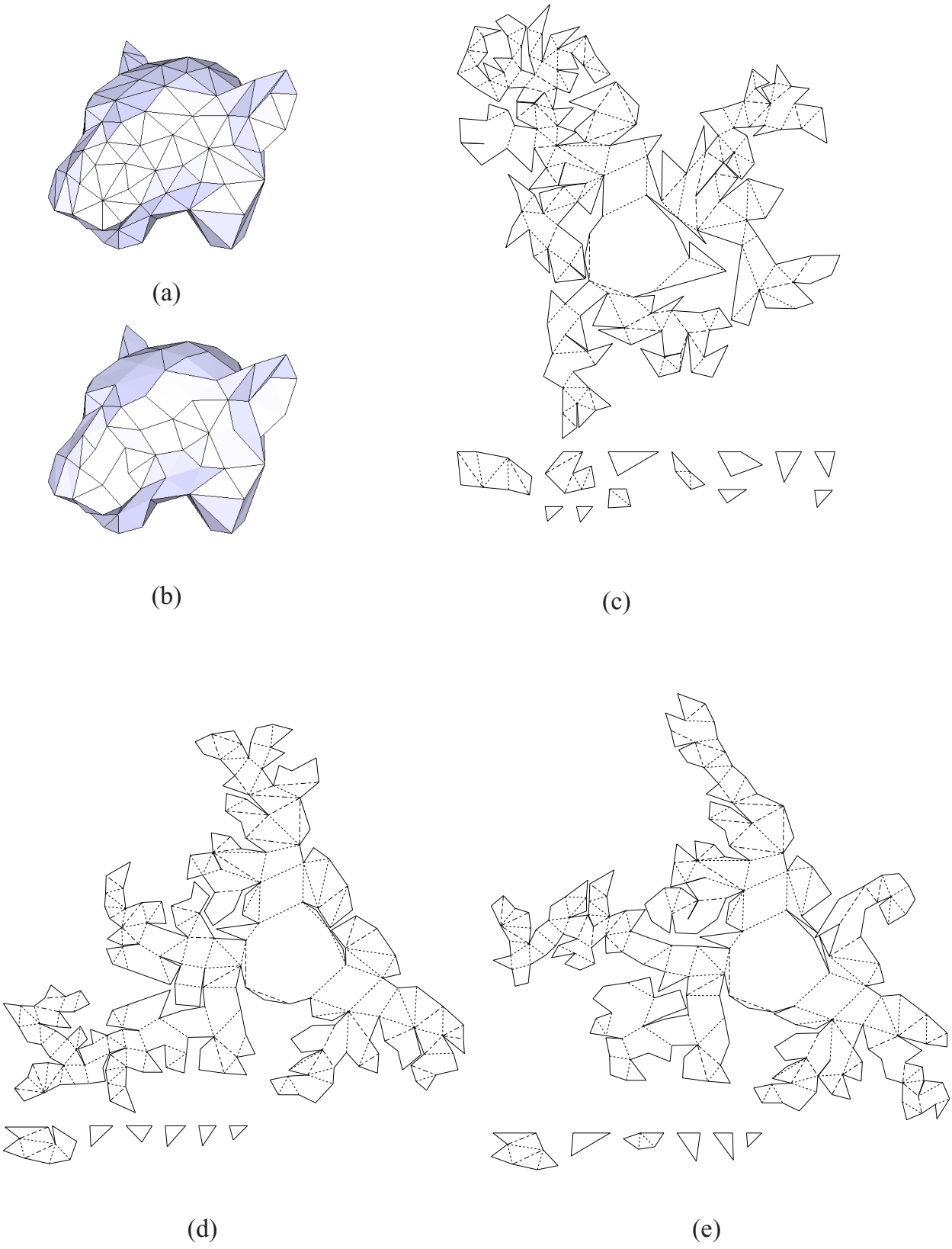


図 3.27: トラの頭部 (面数 246) の展開図
(a) 展開を行ったポリゴンモデル (b) 平面に近い辺を省略表示したポリゴンモデル
(c) ランダム作成した展開図
(d) 辺の長さを考慮して作成した展開図 (e) 辺の長さと角度を考慮して作成した展開図

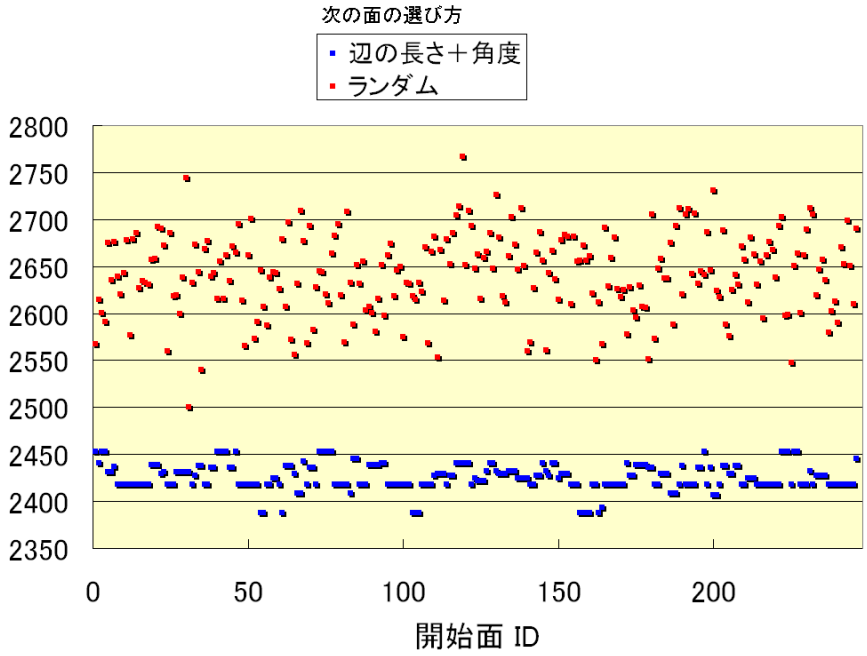


図 3.28: 展開開始面による展開図のコストの違い

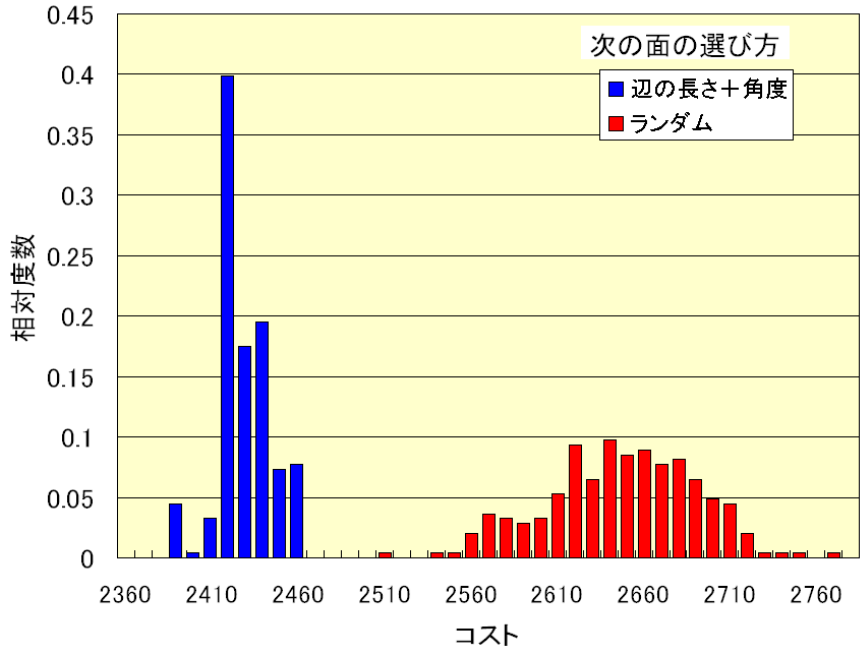


図 3.29: 展開開始面による展開図のコストの分布

3.5　ペーパークラフト用の展開図生成アプリケーション

前節までで、ポリゴンモデルを展開する手法について述べてきた。1 つ 1 つの面を順に展開し、次の面を選択する時に辺の長さなどを考慮することで、ある程度組み立てコストの小さな展開図を得られることが確認できた。

しかし、一般的なペーパークラフトを考えた場合、計算機が自動的に作成した展開図が必ずしも最もよい展開図だとは言えない場合がある。例えば、組み立ての容易さ以外にも、目立たないところに切断線がくるようにしたい、というような要望も考えられる。このような、人間の主観に基づいた要望に、計算機によるアルゴリズムで全て対応するのは難しい。そのため、ペーパークラフト用の展開図を作成するアプリケーションには、人間が予め特定の辺を切断するように指示できるインターフェースや、展開が終わった後でも展開図を編集できるインターフェースがあることが望ましい。

また、実際のペーパークラフトには、のりしろが存在したり、場合によっては貼りあわせ位置を確認できる番号が振られていることもある。一般の人々がペーパークラフト用の展開図を計算機を用いて作成できるようにするには、アプリケーション側でこれらに対応できるようにする必要がある。

3.5.1　展開図編集のためのインターフェース

計算機によって自動で展開図を作成できるようになることで、3 次元形状を容易に紙模型にすることができるようになる。しかし、既に述べたように、1 つの立体にも展開図のパターンは無数にあり、人間が一番よいと思われる展開図を自動で生成することは難しい。そこで、展開図を生成するシステムを開発する上では、ユーザーが意図した展開図を得るためのインターフェースが実装されていることが望ましい。

展開図を構成する各面の形状は、対象とするポリゴンモデルによって既に定められており、これは不変であるから、展開図の形に影響する要素は、面と面の接続関係だけである。従ってユーザーが指示できるものは「どの辺を切断するか」「どの辺を切断しないか」の 2 つである。これらは、実際に展開を行う前に指示してもよいし、展開を行った後で指示してもよい。

展開図生成前のインターフェース

ここでは、展開図を生成する前に、予め「切断する辺」と「切断しない辺」の指定を行うインターフェースを提案する。

「切断する辺」として指定された辺は、図 3.8に記した展開図作成アルゴリズムにおいて、常に「NG フラグ」が立っているものと扱うことで、必ずこの辺が展開図では切断されるようにすることができる。これは例えば動物の展開図を作るときに「首の付け根で胴体と頭部を分割するようにしたい」というようなユーザーの要望があった場合に有効である。

「切断しない辺」の指定は、極端な例として全ての辺を「切断しない辺」にすることはできないので、明示的に決定できるものではない。しかし、図 3.8の (b) の「次に展開する面の決定」において、この辺に接続する面を優先的に選択することで、なるべく展開図上では切断されない辺にすることができる。これは例えば、動物の顔の部分などで「切断されると格好が悪い」というようなユーザーの要望があった場合に有効である。

図 3.30に、実際に「切断する辺」と「なるべく切断しない辺」の指定を行った例と、それによって生成された展開図の例を示す。図中 (a) のオレンジ色で表示されている辺が「切断する辺」として指定した辺であり、黄色で表示されている辺が「なるべく切断しない辺」として指定した辺である。実際にこれらを指定することで、図 (b) のように意図した展開図を得ることができることを確認できる（この例では、切断する全ての辺を明示的に「切断する辺」と指定しているため「なるべく切断しない辺」は指定しなくても結果には影響しない）。

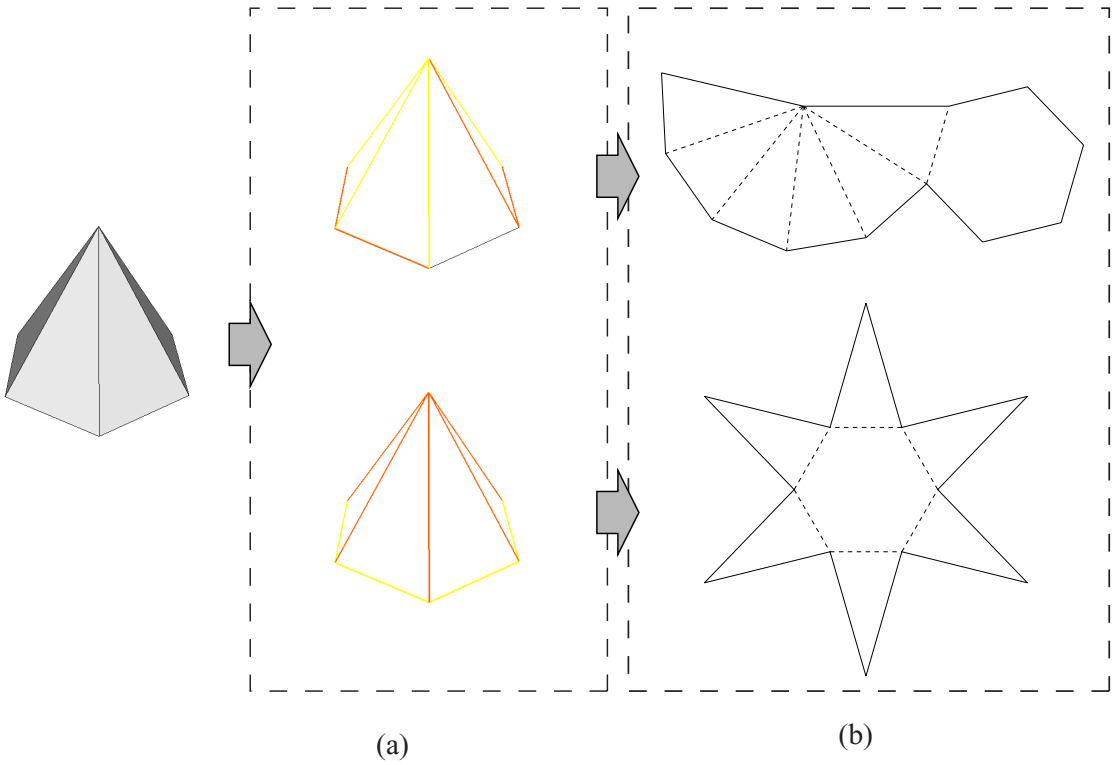


図 3.30: 「切断する辺」と「なるべく切断しない辺」の指定を行った展開

展開図生成後のインターフェース

ここでは、展開図を生成した後で、既に切断されている辺を接続したり、切断されていない辺の切断を行うインターフェースを提案する。これは、実際に得られた展開図を見て、後から修正を加えたい場合に必要なインターフェースである。

切断されている辺を接続するには、展開平面上で一方の面 (A) に、他方の面 (B) を移動させる必要がある。このときに、既に面 B に接続していた面群 (C_n) を、面 B が移動する際にその場に残すのか、それとも B と共に移動するのかを定めておく必要がある。また、この移動に伴って、面の接続関係が変化するため、切断線と折れ線の更新を行う必要がある。

これとは逆に、接続されている辺を切断するには、面の接続関係のみを更新して、折れ線を切断線に変更するだけでよい。特に、面の位置を動かす必要はないが、ユーザーが「分離した」ということを認識しやすいように、互いに位置をずらしてもよい。

また、本質的には工作になんら影響を与えないが、展開図の「見栄え」をよくするためには、展開図の各パーツの回転や平行移動を行え、全体の配置を整えられるインターフェースも必要

である。一般に、ランダムに配置された展開図よりも、 3 次元形状の時の位置関係を把握しやすい展開図の方が、ユーザーにとってはわかりやすいというメリットがある。

図 3.31に、実際に展開図を生成した後に編集を行うインターフェースを実装し、辺の接続と切断を用いて立方体の展開図を対話的に編集した例を示す。なお、編集時には、編集モードを「辺の接続」、「辺の切断」を切り替えて対象となる辺を選択するものとした。

［辺の接続（両側の面が同じパーツに属する場合）］

図 (a) は自動的に展開図が生成された直後に、編集モードを「辺を接続」にし、図中の太線で示される辺を接続する辺として選択した様子である。接続する辺として選択できるものは、「3 次元では隣接する面を持っていて、展開図中では切断されている辺」であり、ここでは赤く表示されている。 1 つの辺を選択すると、それと対応する辺へ向かう矢印が表示され、面の移動先を確認できるようになっている。図 (b) は、実際に辺の接続を行って、それに伴い 1 つの面が移動した結果を示す。

［辺の切断］

図 (c) は前述の (b) の後に編集モードを「辺を切断」にした様子である。切断する辺として選択できるものは、展開図で 2 つの面に共有される辺であり、ここでは赤く表示されている。図中の★印は便宜上後から追加したものであるが、この印の乗る辺を選択した結果は (c) のように、選択した辺で展開図が 2 つのパーツに分離される。

［辺の接続（両側の面が異なるパーツに属する場合）］

図 (e) は前述の (d) の後に編集モードを「辺を接続」にし、図中の太線で示される辺を接続する辺として選択した様子である。 (a)、 (b) では、互いに接続される面が同一のパーツに含まれていたため、 1 つの面のみが移動したが、 (e) の例では、互いに接続される面が異なるパーツに含まれているため、この接続の操作によって 2 つのパーツが 1 つに統合される。このような場合は、接続される面が属するパーツ全体の移動を伴う面の接続を行う。この結果は (f) のようになる。

このようなインターフェースを実装し、展開図の編集を行って、実際に作成したウサギの展開図とその紙模型の例を図 3.32と図 3.33に示す。組み立て時の参考になるように、貼り合わせを行う辺には、互いに同じ数字が表示されるような工夫を行った。

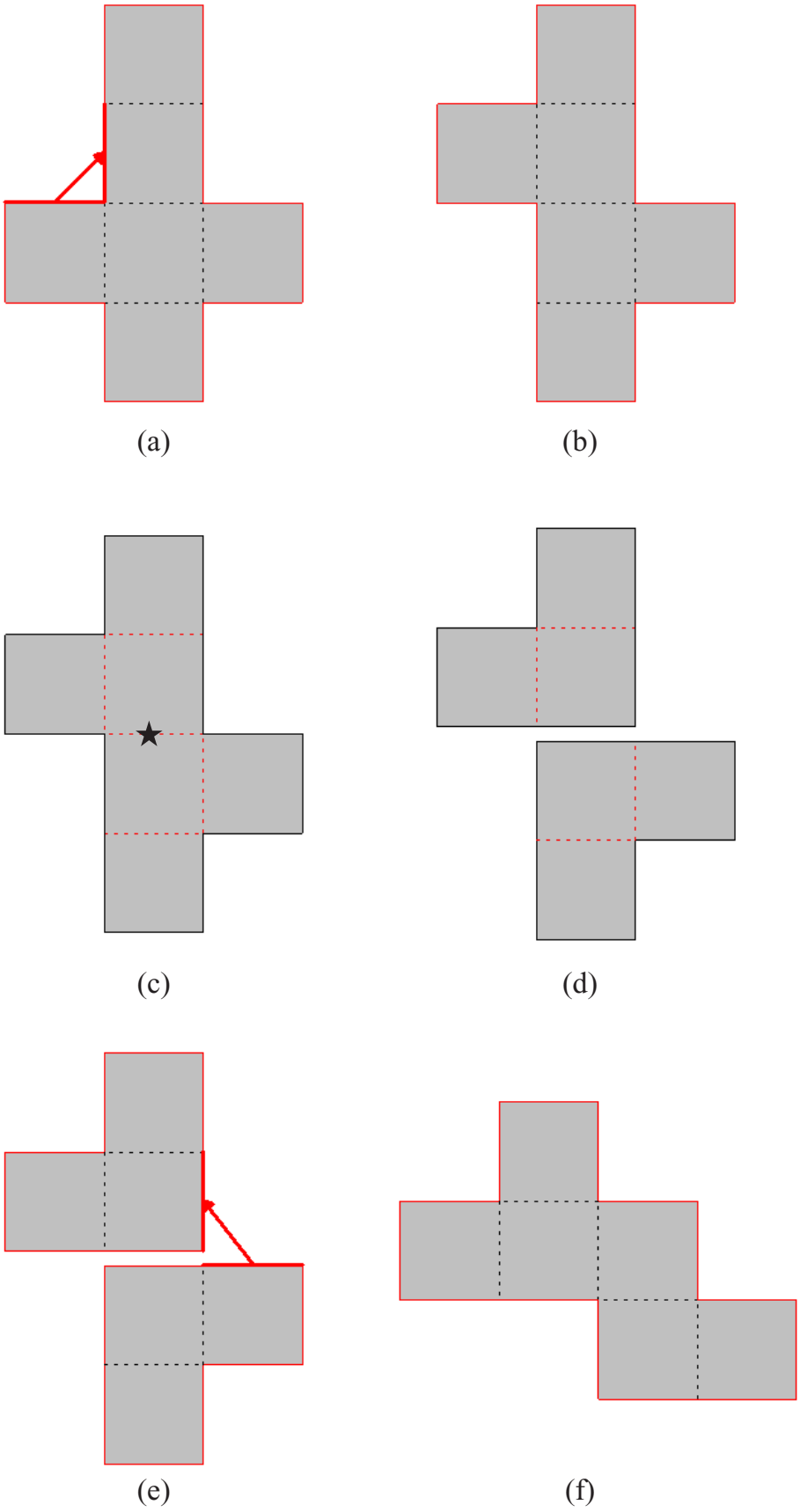


図 3.31: 展開図編集のインターフェース

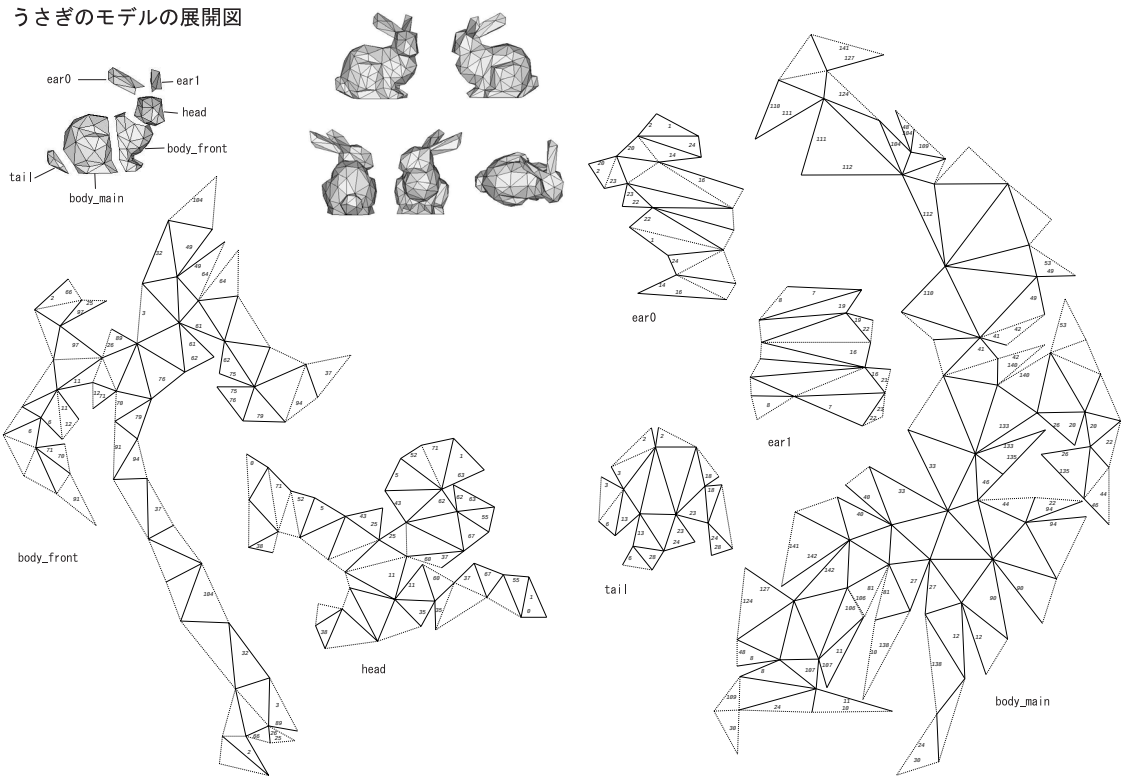


図 3.32: ウサギの展開図の例

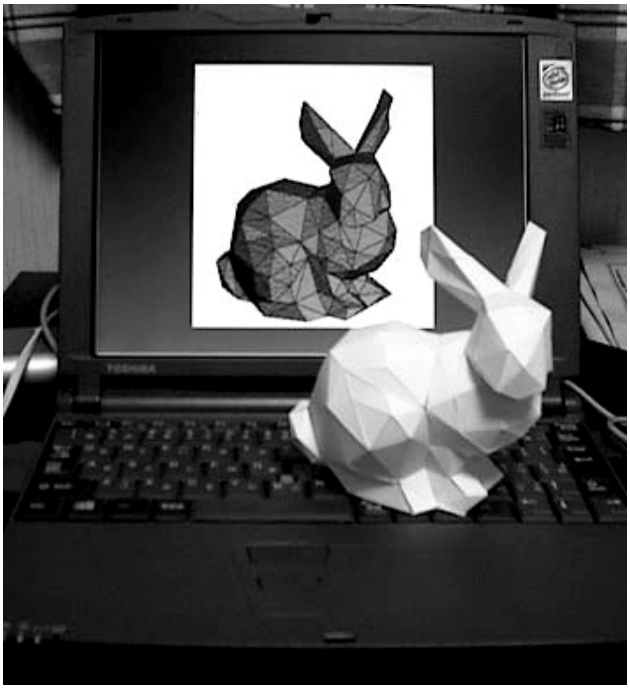


図 3.33: 実際に組み立てたウサギの紙模型

3.5.2　テクスチャの適用とのりしろの生成を施した展開図

前節までに紹介した例はどれも無地の展開図であったが、 CG の世界ではテクスチャを施すことで、少ない面数でよりリアルな形状表現をすることがよく行われる。このようなテクスチャを展開図にも施すことができれば、よりリアルな紙模型を作成することができる。

ポリゴンモデルへのテクスチャの適用は、各頂点にテクスチャマップ用の uv 座標を定義し、テクスチャとして指定された画像の画素を、面が表示される領域の画素に対応付けることで行われる。 3 次元でのポリゴンモデルの面と、展開図内の面は 1 対 1 の対応を持っているため、面の頂点に設定された uv 座標を引き継ぐことで、ポリゴンモデルに施されたテクスチャをそのまま展開図へ反映させることができる。このようにして、容易に立体模型へテクスチャを適用できることは、光造形や切削などによるラピッドプロトタイピングには存在しない紙模型の大きな利点である。

また、前節までに紹介した展開図には、工作の際にのりをつけて貼りあわせを行うための「のりしろ」が存在しなかったが、一般のペーパークラフトの多くにはのりしろが存在する。のりしろは、 3 次元形状の時に 2 面に共有される辺が、展開時に切断されて 2 つの辺になったときに、どちらか一方に生成すればよい。

これらは 3.2.6節の図 3.19で示したように、一般的なハーフエッジ構造の Halfedge 要素を拡張することで対応できる。

図 3.34は、実際にポリゴンモデルの各面にテクスチャを施し、貼りあわせが必要な辺にのりしろの生成を行った例である。

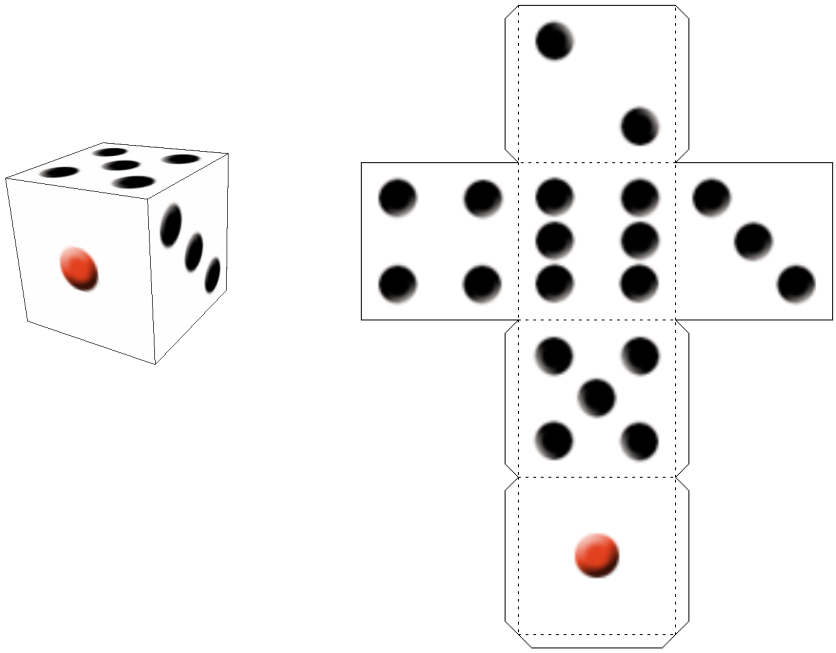


図 3.34: テクスチャの適用とのりしろの生成を施したサイコロの展開図

3.5.3 計算機による工作の支援

これまでに、計算機によってポリゴンモデルの展開図の作成および編集を行う手法を提案してきた。ところで、展開図から紙模型を組み上げる時に、その「組み立て方」がわかりにくく工作のときに手間取ることがある。一般のペーパークラフトでは、展開図と共に「組み立て説明書」が付属することが多い。この理由として、展開図だけが与えられても完成模型がイメージできない、展開図のどこが完成模型のどこに対応するのかがわからない、ということが考えられる。計算機で展開図を作成した場合には、立体に含まれる面と展開図に含まれる面の対応関係が自明であり、また計算機のディスプレイには立体モデルと展開図の両方を表示できることから、これらを有効に活用すれば、計算機によって展開図を作成するだけでなく、計算機によって工作時の支援を行うことも可能だと考えられる。

対応する面の確認

工作のときに組み立て方に戸惑う一番の要因は、展開図と完成模型の対応関係がわからなくなった場合である。これを解決するためには、完成模型のどこの部分になるのかを確認したい面を、ユーザーが展開図上で指定すると、それに対応する面を CG で強調表示する機能が有効である。また、模型の一部分を先に作りたい場合に、これとは逆に CG で表示されたポリゴンモデルの面を選択すると、それに対応する展開図の面が強調表示される機能も有効である。

図 3.35は、この機能を実際に実装した様子である。例えば図 (a) に示す容器の左右側面の形状はまったく同一ではないが、非常に似通った形をしているため、 (b) の展開図では、どちらがどちらの側面に対応するのかを判断するのが難しい。このような時に、対応を確認したい面を展開図で選択したり、立体表示画面で選択することで、互いに同じ面が赤く強調表示されることで、容易に対応を確認できる。

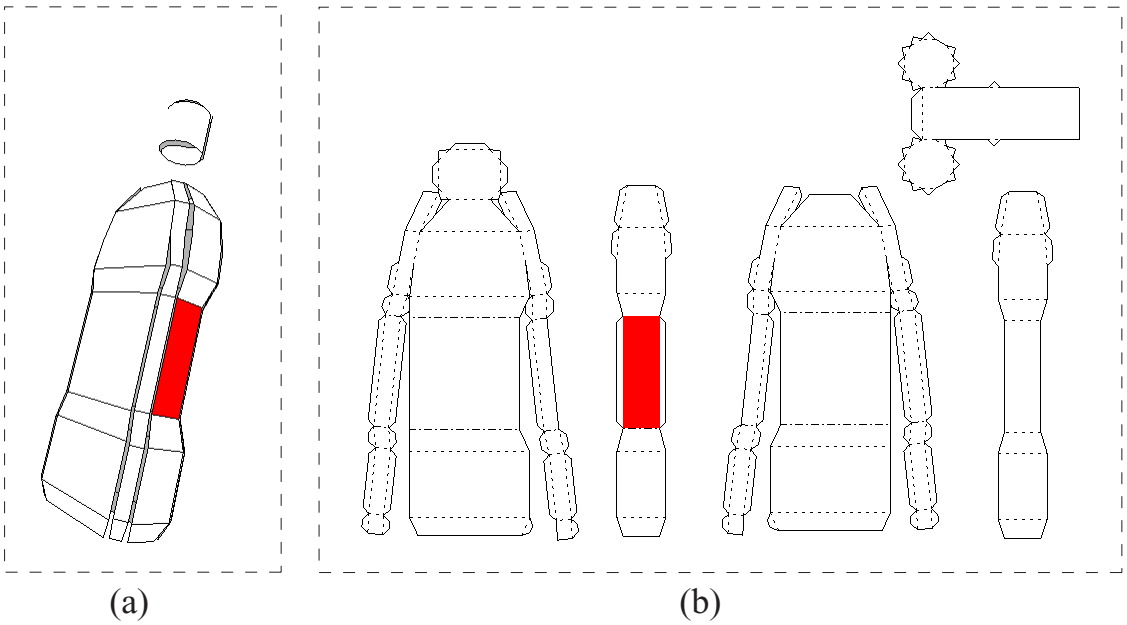


図 3.35: 対応する面の確認

展開のアニメーション表示

立体形状と展開図の対応を確認するためのもう一つの方法として、立体と展開図の中間形状をアニメーションで表示することが考えられる。

2 つの形状をアニメーションによって補間する手法は**モーフィング**と呼ばれ、もっとも単純な補間方法は、2 つの形状で互いに対応する頂点の間を線形補間することである。しかし、この方法を立体形状と展開図の間に適用すると、補間の途中で各面が歪み、面積が一定に保たれない問題がある。実際の紙はもちろん面積が不変であるため、見た目が紙模型の展開とは大きく異なるものになってしまう。

そこで、3.2.2節で述べたような、ポリゴンモデルを構成する面の剛体変換による展開に必要な行列を算出し、その行列に現れる平行移動距離と回転角度を線形補間することで、滑らかなアニメーションを行う手法を提案する。

今までに述べてきた手法によって生成した展開図には、そのパーツの数だけ連結な面グラフ（3.2.1節参照）が存在する。ここでは簡単のために 1 つのパーツだけに着目すると、この面グラフは最初に展開した面 $F_{(1)}$ を起点に、閉路を持たずに連結した「木構造」になっている。このポリゴンモデルから任意に取り出した面 F を考えると、その面 F から初期展開面 $F_{(1)}$ につながる経路はただ 1 つだけ存在し、その間に存在する面の数によって、初期展開面から面 F への深さ n を定義できる。この、深さ n の面を $F_{(n)}$、この面上の点の座標を $\boldsymbol{P}_{(n)}$ とすると、展開平面上におけるこの点の位置 $\boldsymbol{P'}_{(n)}$、次式で表すことができる。

$$\boldsymbol{P'}_{(n)} = \boldsymbol{M}_{(1)}\boldsymbol{M}_{(2)}\boldsymbol{M}_{(3)}\ldots\boldsymbol{M}_{(n)}\boldsymbol{P}_{(n)} \tag{3.8}$$

ここで$\boldsymbol{M}$は回転と平行移動による剛体変換を表す 4 × 4 の同次行列で、特に $\boldsymbol{M}_{(1)}$ は面グラフの最初の面 $F_{(1)}$ を展開図平面上の $F'_{(1)}$ へ写像する行列である。$i(2 \leq i \leq n)$ について、面 $F_{(i)}$ が面 $F_{(i-1)}$ と共有する辺を $E_{(i-1)}$ とすると、$\boldsymbol{M}_{(i)}$ は面 $F_{(i)}$ を辺 $E_{(i-1)}$ を中心に、面 $F_{(i-1)}$ と同一平面に乗るように回転させる行列である（図 3.36）。

アニメーションを行う時間を $t(0 \leq t \leq 1)$ とし、$t = 0$ の時に立体モデル、$t = 1$ の時に展開図になるとすると、点 $\boldsymbol{P}_{(1)}$ は t の値が 0 から 1 に近づくにつれ、徐々に $\boldsymbol{P'}_{(1)}$ に近づくように移動と回転を行えばよい。点 $\boldsymbol{P}_{(i)}(2 \leq i \leq n)$ は、徐々に面 $F_{(i-1)}$ と同一平面に乗るように、辺 $E_{(i-1)}$ を中心とした回転を行えばよい。

$\boldsymbol{M}_{(i)}(2 \leq i \leq n)$ は辺 $E_{(i-1)}$ を中心とした回転を行う行列であり、時間によって変化するパラメータとして、回転角の大きさだけを考慮すればよい。この回転行列について、角 θ だけ回転する行列を $\boldsymbol{M}_{(i)}(\theta)(2 \leq i \leq n)$ と表すこととする。

$\boldsymbol{M}_{(1)}$ については、時間によって変化するパラメータとして考慮すべきものは、行列の移動ベクトル成分$\boldsymbol{T}$の大きさと、xyz 軸の各軸まわりの回転角である。この行列について、$\boldsymbol{T}$方向に s だけ移動し、各軸周りに角 $\theta_x, \theta_y, \theta_z$ だけ回転する行列を、それぞれのパラメータを用いて $\boldsymbol{M}_{(1)}(s, \theta_x, \theta_y, \theta_z)$ と表すこととする。

以上を用いると、時間 t における面 $F_{(n)}$ 上の点 $\boldsymbol{P}_{(n)}$ の位置 $\boldsymbol{P'}_{(n)}(t)$ は、次の式で表すことができる。

$$\boldsymbol{P'}_{(n)}(t) = \boldsymbol{M}_{(1)}(t|\boldsymbol{T}|, t\theta_{x(1)}, t\theta_{y(1)}, t\theta_{z(1)})\boldsymbol{M}_{(2)}(t\theta_{(2)})\boldsymbol{M}_{(3)}(t\theta_{(3)})\ldots\boldsymbol{M}_{(n)}(t\theta_{(n)})\boldsymbol{P}_{(n)} \tag{3.9}$$

ここで $\theta_{(i)}(2 \leq i \leq n)$ は、面 $F_{(i)}$ と面 $F_{(i-1)}$ の成す角を 180 ° から引いた値であり、$\theta_{x(1)}, \theta_{y(1)}, \theta_{z(1)}$ はそれぞれ、面 $F_{(1)}$ を $F'_{(1)}$ に写像する行列の各軸まわりの回転角である。それぞれの行列 $\boldsymbol{M}_{(n)}$

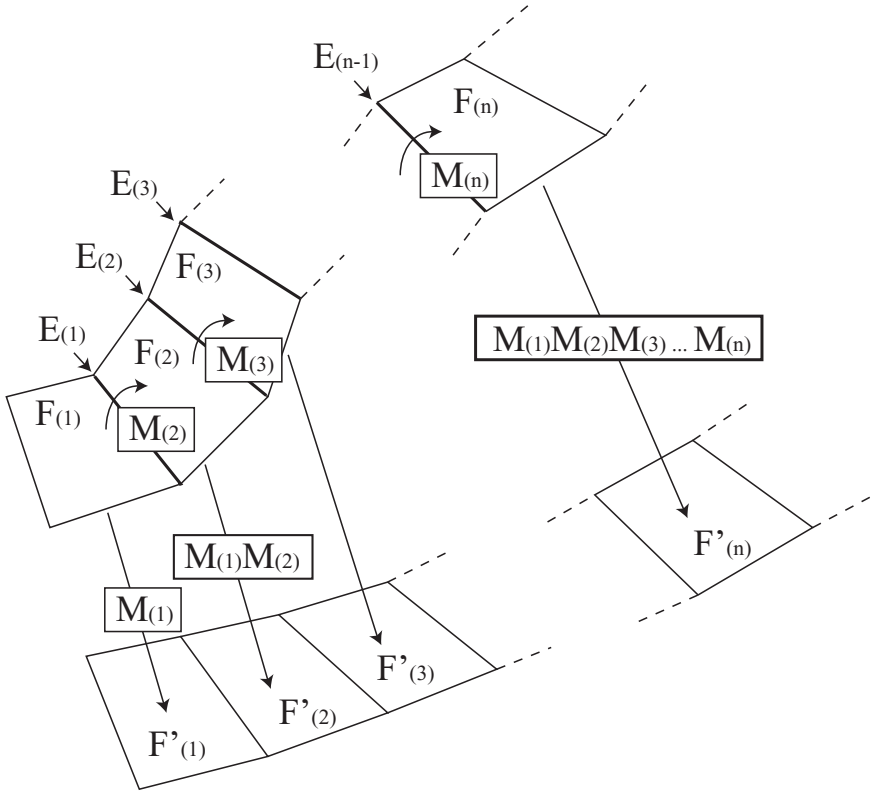


図 3.36: 3 次元モデルと展開図の対応関係

は剛体変換を行う行列であるので、ポリゴンモデルを構成する各面の形状は常に不変であり、$t = 0$ の時には初期状態、$t = 1$ の時には展開された状態となる。

ポリゴンモデルの各面について、t の値を 0 から 1 へ徐々に変化させた式 3.9を適用することで、立体が次第にに展開されてゆく様子をアニメーションできる。また、それとは逆に 1 から 0 へ変化させることで、展開図が徐々に組みあがってゆく表示をアニメーションできる。

このアニメーション表示を実装した結果を図 3.37と図 3.38に示す。ポリゴンモデルが徐々に展開されてゆく様子を視覚的に確認することができる。このような CG アニメーションは、従来の紙媒体のペーパークラフトでは実現できなかったことであるから、計算機を用いることの利点を有効に活用した手法だと言える。

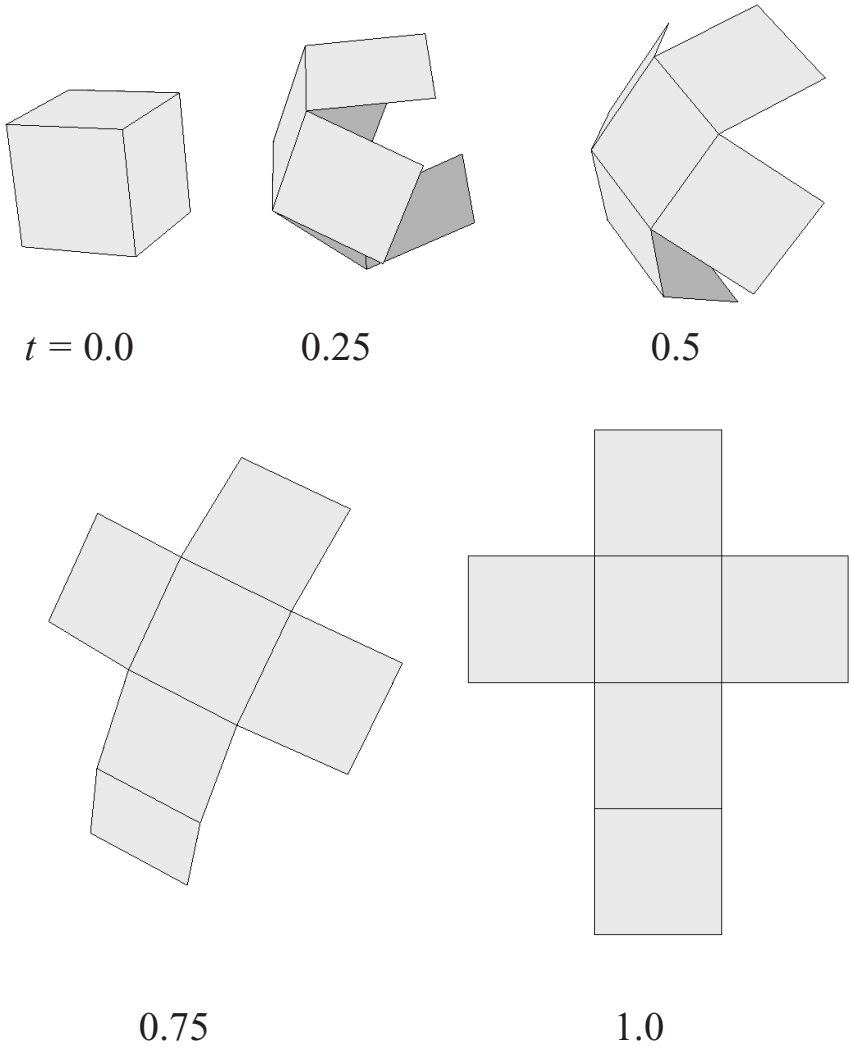


図 3.37: 立方体モデルの展開アニメーション

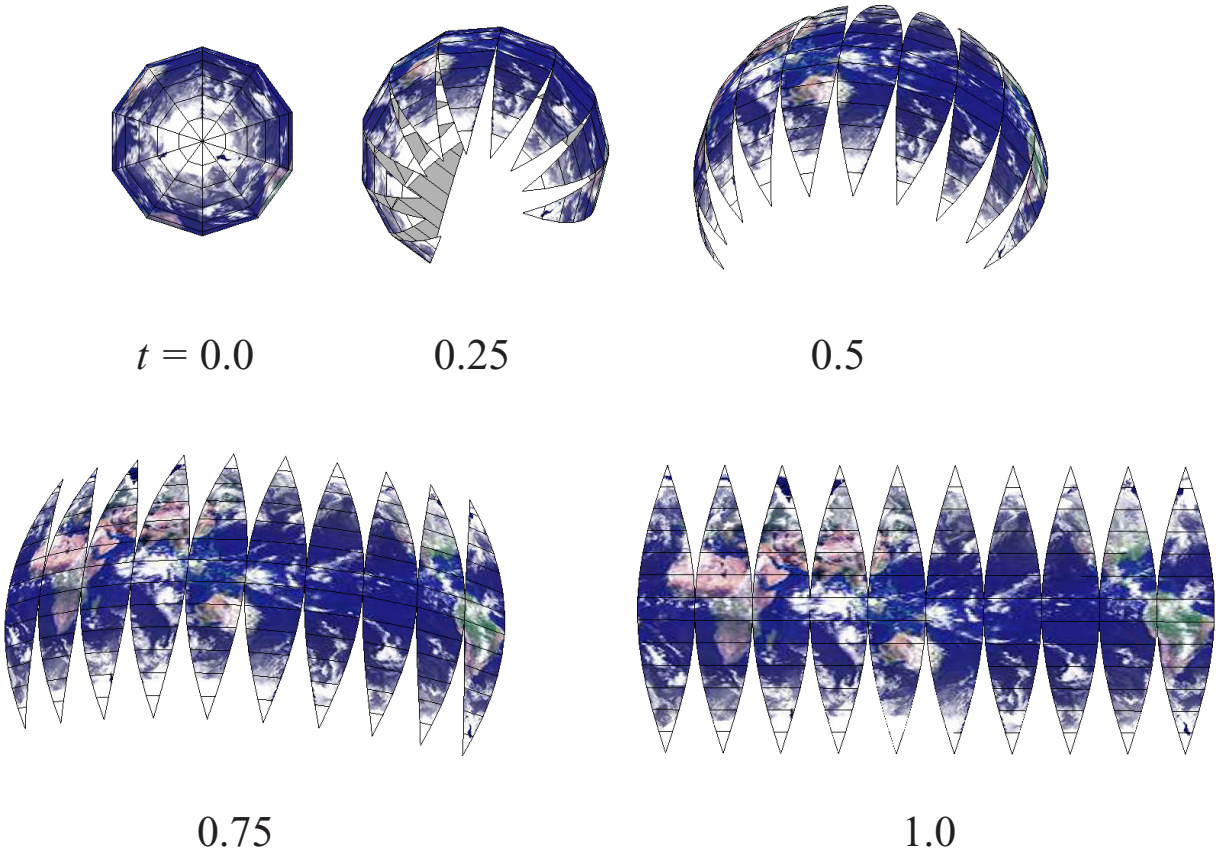


図 3.38: 地球のモデルの展開アニメーション

3.5.4 アプリケーションの実装

本章で述べてきた手法を PC 上に実装し、ポリゴンモデルの展開図作成を計算機によって支援するためのアプリケーションを開発した。立体形状の入力には、 CG の世界で一般的に用いられている AutoCAD[38]、六角大王 Super[39]、メタセコイア [40]、 3D Studio Max[41] などの形式をサポートした。

図 3.39はこのアプリケーションのウィンドウ画面である。計算機内に構築したシャンプーの容器のポリゴンモデルを展開し、その展開図を表示した様子を表している。図に示すように、アプリケーションの左ウィンドウには 3 次元のポリゴンモデルが表示され、右ウィンドウには展開図が表示される。展開の前に「切断する辺」「なるべく切断しない辺」の指定を行え、展開後は 3.5.1節に記したインターフェースで、展開図の編集が行えるようになっている。また、工作時のためにポリゴンモデルの面と展開図の面の対応を両方のウィンドウで確認できるようになっている。

展開図の各パーツのレイアウトについては、 2 次元上での多角形の詰め込み問題として各種の研究がなされているが [42, 43]、ここでは単純にパーツの外接四角形を隅から並べ、その後のユーザーの作業によって適切に配置しなおすこととしている。

このシャンプーの容器の例では、ポリゴンモデルの入力から図 3.39のような展開図を得るまでに要する作業時間は数分程度である。後は展開図をプリンタで印刷することで、実際に工作することができる。

なお、このアプリケーションは、現在エービーネット株式会社 [44] よりペパクラデザイナー [45] という名称のソフトウェアとして市販され、解説本 [46] も出版されている。

このアプリケーションを用いて様々なペーパークラフトが作成され、現在では多くの作品がインターネット上で公開されている [45]。図 3.40,3.41は、 Web 上で公開されている作品の一例であり、従来熟練者の手による試行錯誤によって作られてきたペーパークラフトに遜色のないものが作成できることを見て取れる。

従来、与えられた展開図を組み立てることが多かったペーパークラフトの世界において、手軽にオリジナルの展開図を作成する手段として、本章で提案した手法が大きく貢献できるものであることを確認できる。

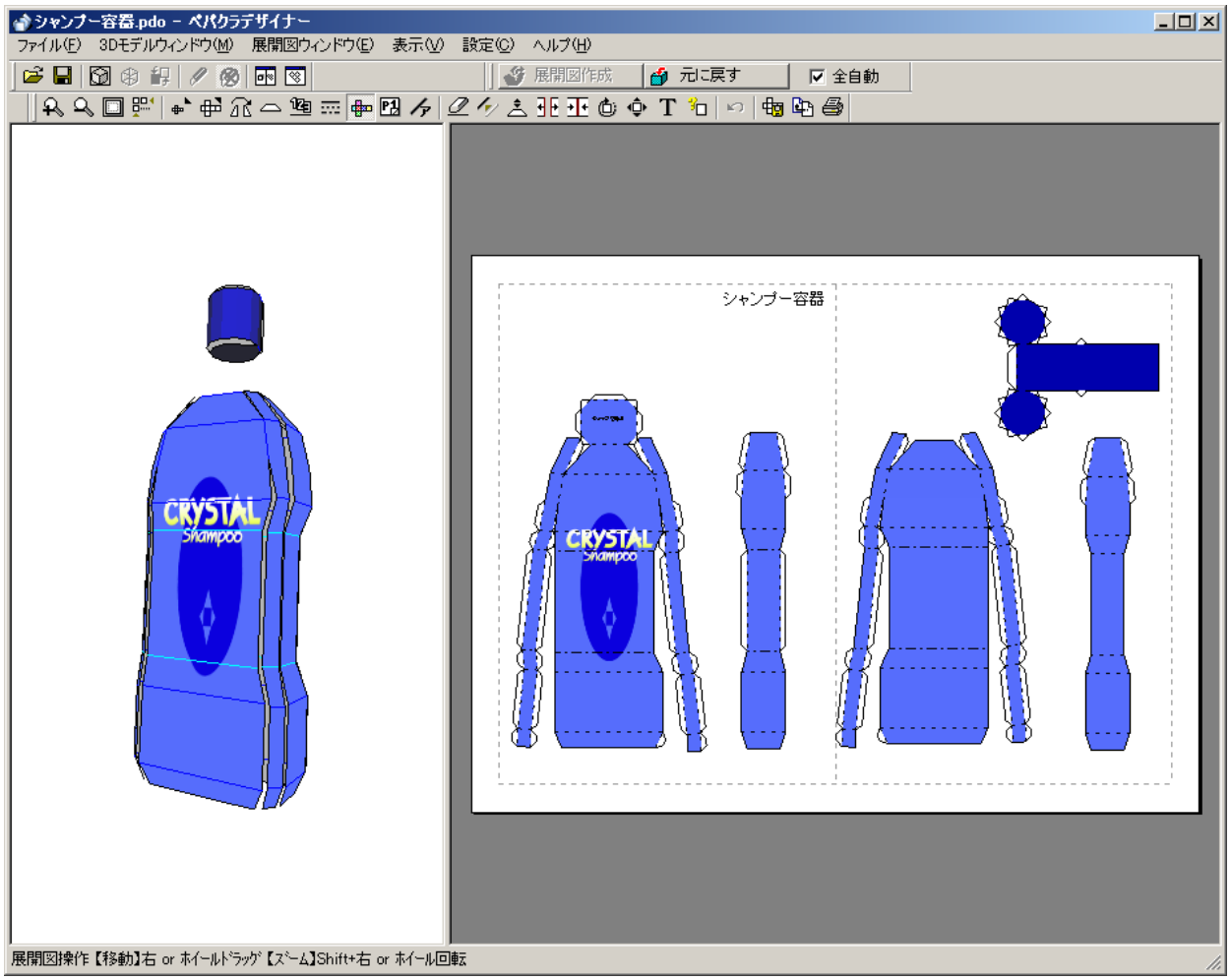


図 3.39: アプリケーションウィンドウ

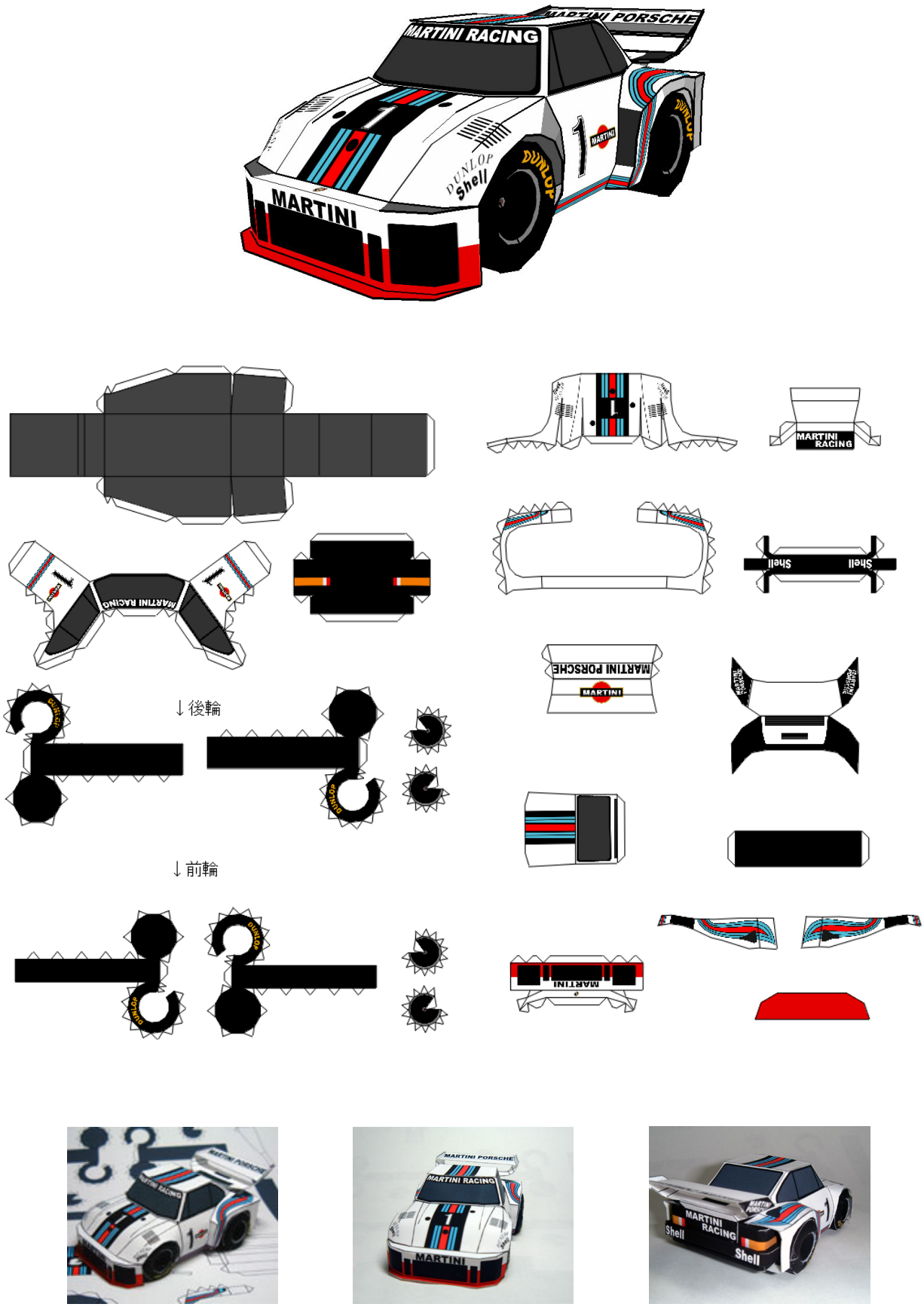


図 3.40: 車の展開図
上段： 3DCG 中段：展開図 下段：完成写真（(c) 鬼屋氏 [47]）

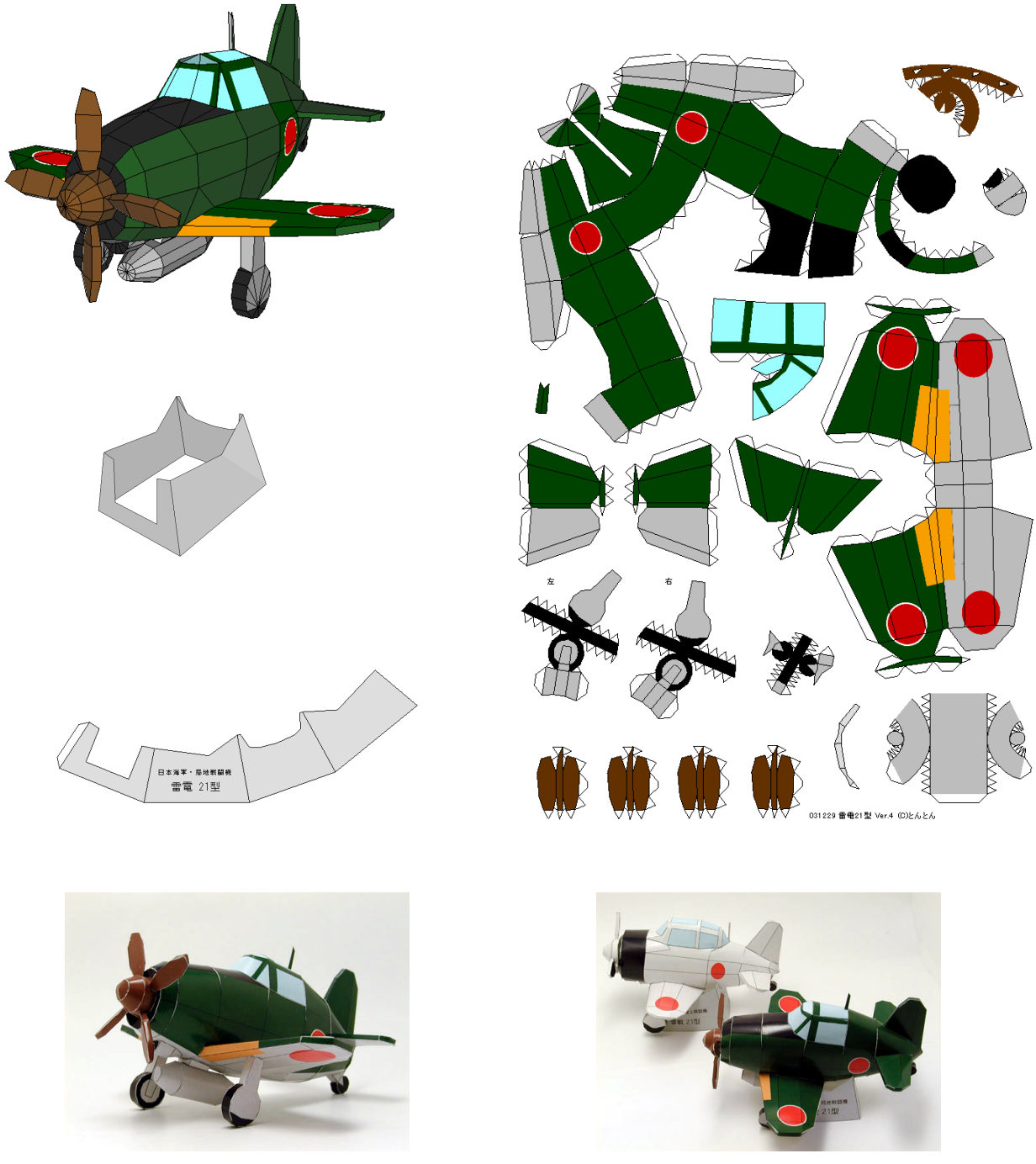


図 3.41: 戦闘機の展開図
左側： 3DCG 右側：展開図 下段：完成写真（(c) とんとん）

3.5.5 考察

ペーパークラフト用の展開図を生成するアプリケーションを PC 上に実装し、その有効性を確かめた。CG の世界で一般的に用いられている形式を入力としてサポートすることで、従来の CG ソフトで作成された立体形状から、ペーパークラフトにするための展開図を作成できるようになった。CG ソフトで立体形状を作ることが純粋なホビーとして普及している現在、そのデータを手にとって触れる形にできることは意義のあることだと思われる。

このソフトウェアは、現在多くの方に使用していただき、実際に様々なペーパークラフト作品が Web 上で公開されている。展開図を編集するためのインターフェースが実装されたことで、計算機が生成した展開図を好みに応じて編集できることが高く評価されている。また、展開図と立体上の面の対応を確認できる仕組みを利用することで、従来は不可欠であった組み立て説明図が無くても画面を見ながら貼り合わせ箇所を確認できるようになった。

しかし、本格的なペーパークラフトを扱いと考えているユーザーからは、展開図上の辺に対応する線分のスタイルや色を自分で定義したいという要望や、本来なら位相的につながっていない箇所に強制的にのりしろを生成したり、自動生成されたのりしろを個別に削除できるようにして欲しいなどの細かい要望が多数上がった。これらは、今後のアプリケーション改良の課題となるであろう。また、展開図を生成した後で、その様子を見てから、例えば「細い隙間は面の形を変形させてでも無くしたい」と感じた場合に「立体形状の編集からやり直さなくてはならないのが大変」という意見もあった。しかし、展開図を編集した場合に、その影響が対応する立体にどのような影響を表すのか、そもそも位相を保持しながらの変形は可能であるのか、というのは非常に難しい逆問題である。それとは別に、立体形状を操作することで、リアルタイムに展開図の形状の変化の様子を確認できるようにすることは、ある程度の難易度はありながらも、ローカルな範囲での展開図作成と、それが他へ及ぼす影響を考慮することで、おそらく実現可能であろう。

また、本研究では、展開図の良し悪しには「組み立てやすさ」のみを考慮していたが、実際には「見た目の美しさ」がかなり重視されることがわかった。例えば左右対称な立体形状の展開図は、やはり左右対称な形になっていることが望まれる。また、人や動物の顔などの展開図は、顔の部分は展開図上でもきちんと上下を維持していることが望まれる。前者は展開図を作成するときの問題であり、後者は展開図を配置する時の問題である。人間が見て、「良い」と思う展開図とはどのようなもので、それはどのようにしたら計算機で生成できるかについては、まだまだ研究の余地がありそうだと思われる。

展開図の作成には直接関係無いが、一般的な CG ソフトでは「見た目さえよければ良い」という発想に基づいていることが多く、面のつながりなどの位相はあまり考慮されていないことが多い。例えば、面と面が干渉していたり、宙に浮いている部分があるような形状は、そのままペーパークラフトにすることができない。また、平面上に乗らない 4 点以上の点から構成される面や、面積がゼロの面など、数値計算の上で問題のあるデータも散見された。これらに対応するためには、データ補正を行う仕組み、もしくは始めからペーパークラフト用の立体形状データを作ることを目的としたモデリングソフトウェアを開発することなどによる対応が考えられる。

第 4 章

メッシュモデルの近似展開図の作成

第 4 章

メッシュモデルの近似展開図の作成

本章では、面の数が多くそのまま展開した場合には工作するのが現実的ではないようなメッシュモデルに対して、近似的な展開図を作成する手法について述べる。はじめに従来のメッシュモデルの展開方法と簡略化手法について紹介した後、新しく提案する手法の概要を述べる。その後、手法の詳細について延べ、その手法を面の数が約 2 万程度のメッシュモデルに適用し、実際に滑らかな曲面部を持つ紙模型を作成した例を紹介する。ここで提案する手法は一般的なメッシュ簡略化の手法とは異なり、 STRIP と呼ぶ帯状の形状の集合でメッシュモデルを近似する点に特徴があり、この手法で作成する展開図では、紙の柔軟性を活かした滑らかな紙模型を作成することができる。

4.1 メッシュモデル

平面多角形の集合から構成されるポリゴンモデルの中で、特に多くの面で曲面を近似したモデルを**メッシュモデル**と呼ぶ。どの程度面の数が大きいものをメッシュモデルと呼ぶか明確な定義は存在しないが、ここでは数千以上の面を持つものをメッシュモデルと呼ぶこととする。近年では、計算機の処理速度の向上と記憶容量の増大、および 3 次元測定器による実物モデルのスキャニング技術の普及により、数十万、時には数百万もの面を含むメッシュモデルが扱われることがある。

メッシュモデルはポリゴンモデルの 1 つであるから、前章で述べた手法を用いてその展開図を作成できる。CG の分野では曲面をメッシュモデルで近似することは一般に行われているため、可展面でない曲面も、このようなメッシュモデルで近似表現することで、任意の曲面を近似的に展開できることになる。

しかし、数千を超える面を含むメッシュモデルをそのまま展開した場合は、あまりに複雑な展開図ができてしまい、これを実際に紙模型として組み立てることは現実的でない。図 4.1は、面の数が 4624 のサイのメッシュモデルであるが、この程度の面数であっても、もはや紙工作には適さない展開図となってしまう。

そこで本章では、このような面の数が多いメッシュモデルを紙模型にするために、紙の特徴を考慮した面の簡略化を行って、近似的な展開図を作成する手法を提案する。

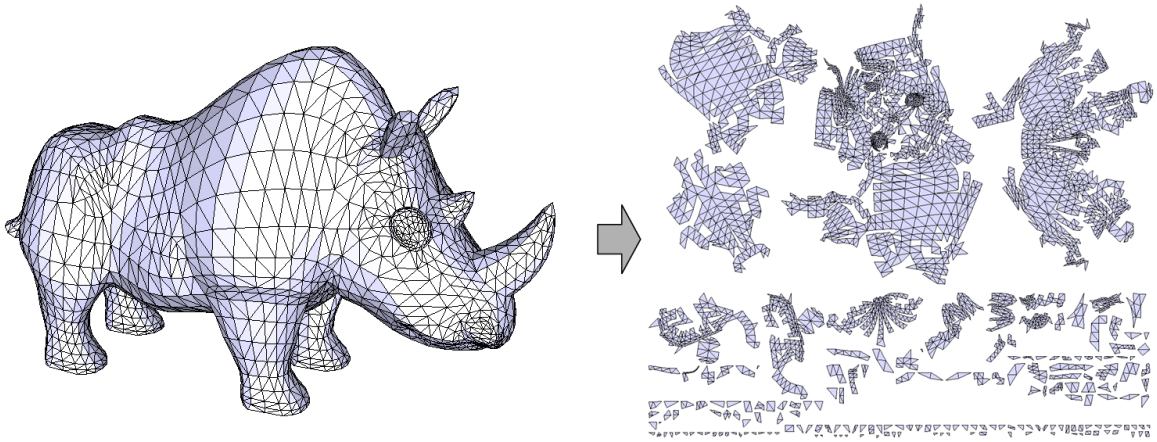


図 4.1: サイのメッシュモデル（面数 4624）の展開図

4.2 メッシュモデルの展開

CG の分野では、 3D データにテクスチャマッピングを行う際に、各面の UV 座標を決定するための展開図が必要となるため、メッシュモデルを平面に展開する手法については Sorkine ら [48]、 Lévy[49]、 Sheffer[50] などの、多くの研究がなされている。

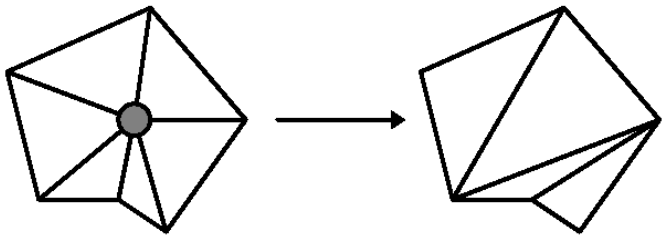
しかし、このテクスチャマップ用の展開を行う場合は、レンダリング時の画像のギャップを抑制することが優先されるため、面の形を歪めることで裂け目や重なりの無い展開図を得ることが行われる。立体形状に含まれる面と展開図上の面の形状が異なるため、今まで議論してきた「展開図」とは本質的に異なるものである。テクスチャマッピング用の展開の研究は、面の歪みを小さく抑えることに重みを置いているものもあるが [51]、たとえ程度が小さくても、歪みを含む展開図を紙模型に用いることはできない。

このように、曲面や、その曲面を近似したメッシュモデルに対して、それを紙模型にするための展開図を作成する手法は未だ確立していない。

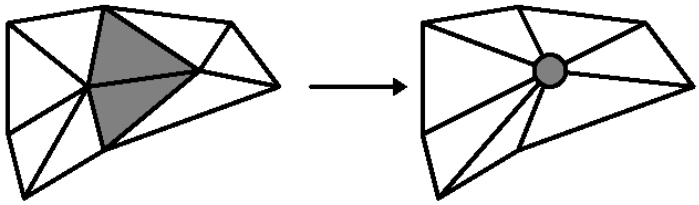
4.3 メッシュモデルの簡略化手法

メッシュモデルに含まれる面をそのまま展開しても実際には展開図が複雑すぎて工作できないため、その解決策としてモデルに含まれる面の数を減らすことが考えられる。メッシュモデルの面の数を減らすことは、メッシュの**簡略化**と呼ばれ、データ量の軽減や処理速度の高速化に有効であることから、 CG の世界では古くから研究されてきている。

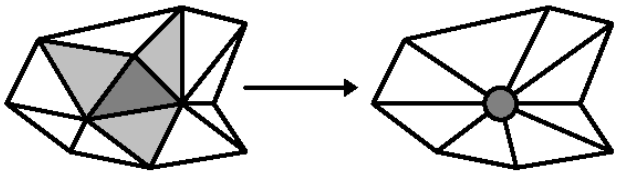
メッシュの簡略化の手法は、文献 [52][53] で多数紹介されているが、いずれの手法も基本的には図 4.2に示す頂点、エッジ、面というメッシュの構成要素を 1 つずつ順に削除する処理（それぞれ、 Vertex remove, Edge collapse, Face collapse と呼ぶ）を繰り返すことで面の数の削減を行っている。



Vertex remove operation



Edge collapse operation



Face collapse operation

図 4.2: 簡略化のローカルオペレーション

メッシュの簡略化においては、面の数を減らしながらも元の形状をなるべく維持する近似精度と、その処理の計算コストにおいてトレードオフの関係がある。

数ある簡略化手法の中で、 Garland ら [54] はバーチャルエッジという仮想的なエッジを導入することで Edge Collapse を一般化させ、その処理ごとにオリジナルメッシュモデルとの誤差を最小にする位置へ頂点を移動させることで、精度が高く処理速度の速い簡略化手法を提案した。この手法を実装した QSlim[55] というアプリケーションはメッシュ簡略化のツールとして広く用いられている。このツールを用いてメッシュモデルを簡略化した例を図 4.3に示す。

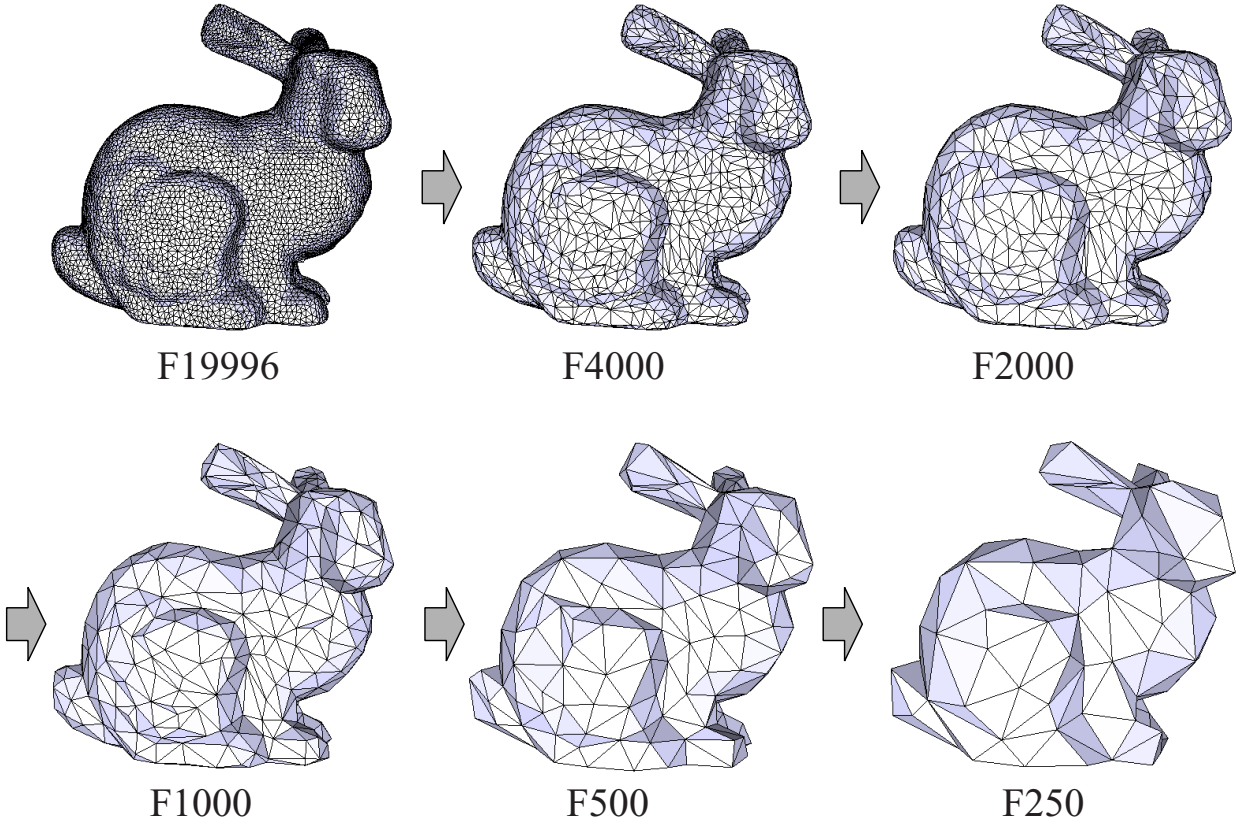


図 4.3: QSlim を用いたメッシュ簡略化の例

4.4 手法の概要

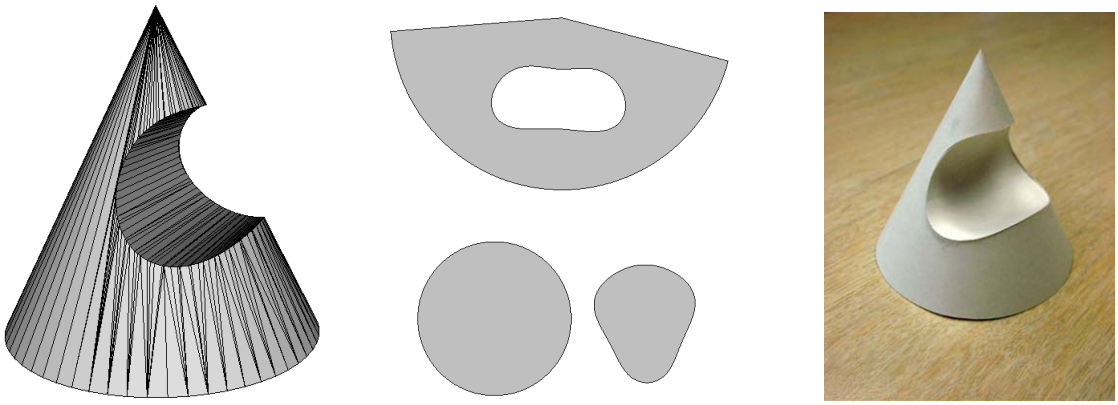
前節で述べたように、既存のメッシュの簡略化手法を用いることで、工作できない程に面の数が多いメッシュモデルも、工作が可能な程度まで面の数を減らすことができる。しかし、これらの手法でメッシュの簡略化を行った場合、細かい面の集合で表現されていた滑らかな形状は、角ばった辺を持つ荒い面の集合になってしまう。この簡略化モデルを紙模型で作成した場合、元の全体的な形状特徴は維持できても、「滑らかさ」という特徴は失われてしまうという問題がある。

ところで、3.3節でまとめたように、紙模型の工作にかかるコストは単純に「面の数」に依存するものではないことがわかっている。また、従来の熟練者の手によって作成された紙模型には、紙の柔軟性を活かした滑らかなモデルが多い。

そこで本章では単純に面の数を減らすだけではなく、紙の柔軟性を活かすことで、元の形に近い滑らかなモデルを低い工作コストで作成できる展開図を作成することを目的とし、そのための手法を提案する。

4.4.1 STRIP

紙模型の工作にかかるコストは3.3節で述べたように、「切り抜きコスト」「折り曲げコスト」「貼りあわせコスト」の 3 つの要素から成る。切断される辺の総延長が短かければ「切り抜きコスト」と「貼りあわせコスト」を軽減することができ、折り曲げる辺が少なければ「折り曲げコスト」を軽減することができる。ところで、折り曲げる辺は、一般に展開図上で 2 つの面に共有される全ての辺であるが、この辺を共有する面の 3 次元での成す角が平面に近い場合、紙の柔軟性を活かして折り曲げ操作を省略することができる。例えば図 4.4のように、円柱や、円錐などの可展面を細長い面の集合で近似した場合、各辺の成す角は平面に近いため、紙の柔軟性を活かすことで、ポリゴンモデルの各辺を折らずに作成できる。



(a) (b) (c)

図 4.4: 円錐面と円柱面からなる形状
(a) ポリゴンモデル（面数 352） (b) 展開図 (c) 紙模型の写真

また、一般に多角形面の集合からなる 3 次元形状を展開する場合、どこかに切り込みを入れ

なくてはならないが、図 4.5のような連続した三角形面によって構成され、内部に頂点を持たない帯状のモデルである場合、切り込みを入れずに平面に展開できる。このような形状は、グラフィックス API[56] において Strip と呼ばれる。これらの、 Strip は、ある三角形を介して互いに接続することができ、この場合も内部に頂点を含まないので、容易に展開することができる。また、内部にループを持つ場合は、一部に切り込みを追加することでそのループを除くことができる（図 4.6）。以降では、このような Strip、および Strip が複数接続した形や、ループを持つ Strip を、単に **STRIP** と呼ぶこことする。

また、図 4.4の例のように、各辺の成す角が平面に近い場合は、紙の柔軟性を利用することで折り曲げ作業を省略できるため、面の数が多くても工作のコストが小さく、さらに紙の柔軟性を活かした滑らかな形状を表現できるという利点がある。

そこで本手法では、対象とするメッシュモデルを滑らかな STRIP の集合で近似表現することで、工作のコストの軽減と、滑らかな形状の作成を実現することを目指す。

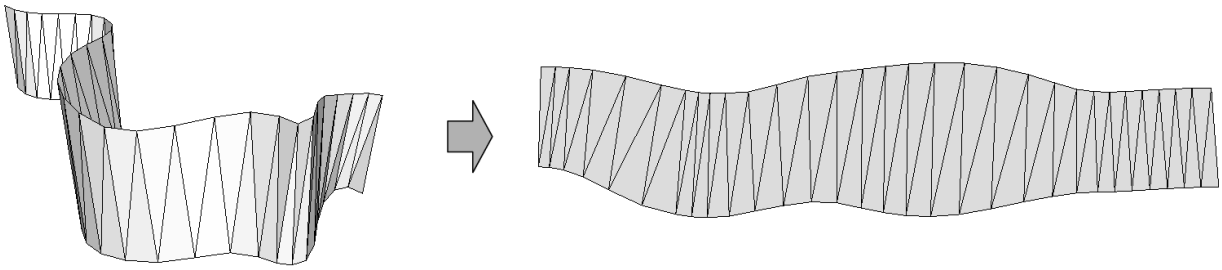


図 4.5: 単純な STRIP とその展開図

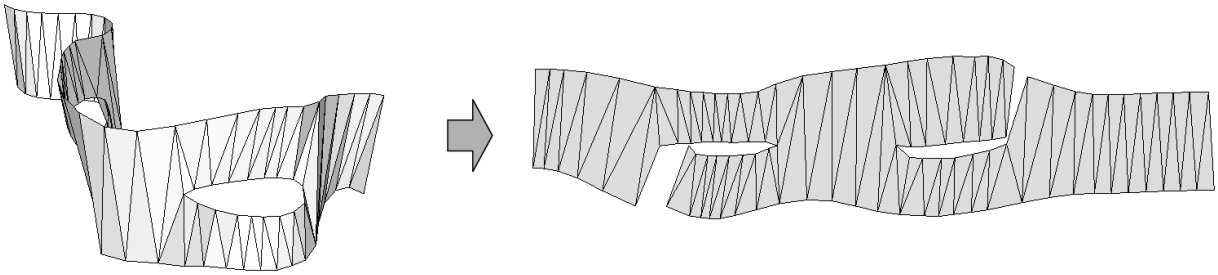


図 4.6: ループを持つ STRIP とその展開図

4.4.2 手法の流れ

本手法の流れは、以下に示す Step1 から Step6 のようになる。まず切り込みの入る場所となる切断線を決定し（Step1 ～ 4）、この切断線を残しながらメッシュモデルの簡略化を行うことで STRIP の集合を作成し（Step5）、その簡略化モデルの展開図を生成する（Step6）。なお、本手法では簡単のために、モデルを構成する各面は三角形である**三角形メッシュモデル**で、位相が円盤または球と同位相のものを対象とする。

Step 1 パーツ分け

まず最初に、対象とするメッシュモデルのパーツ分けを行う。これは、例えば動物のモデルを展開する場合、頭、胸、腕、脚という、その形状特徴に基づくパーツ分けを行ってから、その後で個々のパーツを展開することを考えた方が工作しやすい展開図を得ることができるためである。

Step 2 STRIP 領域の生成

Step1 で得られた各パーツに対して、含まれる三角形の境界からの距離に基づいた領域分割によって、帯状の形をした **STRIP 領域**の集合に分割する。ここで生成する STRIP 領域は、後の工程で 1 つの STRIP に置き換えられる。

Step 3 切断線の追加

パーツの境界と STRIP 領域の境界を展開の際の切断線とする。また、元のメッシュモデルの形状特徴をなるべく維持できるように、 Step1 のパーツ分けの際に生成した特徴線と、円盤と同位相の STRIP 領域の中心に新しい切断線を追加する。

Step 4 切断線の平滑化

Setp2 で生成した STRIP 領域の境界と Step3 で追加した切断線をラプラシアンオペレーターを用いて平滑化する。これにより、展開図の微小な折れが無くなり工作の際の切り出しを行いやすくする。

Step 5 簡略化

Step2 で生成した STRIP 領域の境界と、 Step3 で追加した切断稜線の上に乗る頂点だけを残し、 Garland ら [54] の手法による Edge collapse と、 Vertex decimation の手法を用いたメッシュの簡略化を行う。この結果得られるメッシュは、境界以外に頂点を持たない STRIP の集合となる。

Step 6 展開

Step5 の簡略化によって、 STRIP の集合で近似されたモデルを、前章で述べたポリゴンモデルの展開図作成手法で平面に展開する。

以降の節では、これらの各ステップについて詳細を述べる。

4.5 STRIP 集合による近似展開図の作成

本節では、前節に記した手法の流れの各ステップについて、それぞれの具体的な内容を記す。

4.5.1 パーツ分け

本手法ではまず始めに、与えられたメッシュモデルを、その形状特徴に基づいて大まかにパーツ分けする。

メッシュモデルを複数のパーツ（領域）に分割する手法は、テクスチャマッピングやメッシュ簡略化、マルチレゾリューション表現、形状認識などの分野で必要とされ、様々な研究がされている。

テクスチャを生成する際には、なるべく歪みのないマッピングを行うための領域分割や、領域の境界が目立たない箇所になるような配置が要求される。このような領域分けの手法には、Lévy[49]、 Sander[51] などの研究がある。

メッシュ簡略化やマルチレゾリューション表現を行う際には、特定の領域に属する複数の面を 1 つの面で近似するため、元の形状特徴を維持するような領域分けが必要となる。このような目的の領域分けについては、 Garland[57] や Eck[58] などがある。

また、形状認識の分野では、 3 次元計測器によって生成されたメッシュモデルをその特徴に応じて領域分けを行うことがなされており、 Djebali[59] や Trucco[60] などの研究がされている。

さらに近年では、与えられたメッシュモデルに形状制御のための骨格を定めるための領域分けの研究 [61] も行われている。

これらの各手法には、それぞれの特徴があるが、本手法では Lévy の手法を用いることとした。この手法は、メッシュモデルにテクスチャを貼る際に必要となるテクスチャマップ生成用の領域分けを行う手法で、鋭角な稜線を含む特徴的な部分や、近傍よりも曲率の大きなところに領域の境界を配置することで、繋ぎ目が目立ちやすい滑らかな部分には領域の境界が現れにくい、という特徴がある。例えば文献 [49] の例題として紹介された図 4.7では、ウサギの首、耳、尻尾の付け根が領域の境界となり、全体を複数の特徴的な領域に分けることが実現されていることがわかる。このような領域分けは、紙模型用のパーツ分けにも利用することができる。ただし、この文献中で紹介されている展開図の生成手法はテクスチャマップのためのものであるため、面の歪みを含んだものであり、紙模型にそのまま用いることはできない。

特徴線の抽出

Lévy の手法では、まず特徴線の抽出を行い、その特徴線が境界となるような領域分けを行う。

特徴線は、メッシュに含まれる全てのエッジの中から鋭角なものを一定の割合だけ抽出し、それを伸ばしてゆくことで得られる。なお、鋭角さを示す尺度には隣接する面の法線間の角度（SOD:second order differences [62]）を用い、特定の割合のエッジが抽出されるような閾値を使用する。

この手法のアルゴリズムは図 4.8に示すとおりであり、処理の概要は次のとおりである。この手法には、ノイズによる微小な特徴線を除去できるという特徴がある。

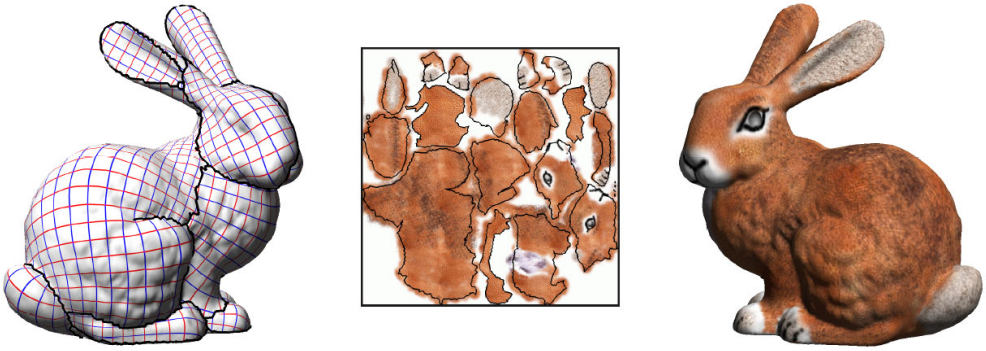


図 4.7: Lévy の手法によるテクスチャマップ生成用の領域分け
（Lévy[49]）

- メッシュモデルに含まれるハーフエッジ群の中から、 SOD の大きなエッジに付属する一続きのハーフエッジ列（図 4.8中の S）を深さ優先探索で生成する。
- 鋭角さが閾値よりも大きいハーフエッジ列が見つかったら、その先頭を特徴線（図 4.8中の $detected\_feature$）に追加する。
- 曲率の高い領域に多くの特徴線が集中するのを避けるために特徴線の隣接エッジにはタグ付けを行って回避する。

図 4.8のアルゴリズムの中には、 max_string_length と $min\_feature\_length$ の 2 つの定数が現れる。前者の値は特徴線を抽出するときの探索の深さを表し、値が大きいほど互いに離れた鋭角エッジが連結されて 1 つの特徴線が長くなりやすくなる。後者は抽出された一連のエッジが、特徴線として認められる長さの閾値を表し、この値が小さいほど短い特徴線も許容されるようになる。

同じメッシュモデルにこのアルゴリズムを適用した場合も、この定数の値によって抽出される特徴線の結果が異なるため、望ましい抽出結果を得るには、値を変えながらの試行錯誤が必要となる。この手法を面数が 19,996 のウサギのメッシュモデルに適用した結果は図 4.9のようになった。図の (a) 〜 (c) は、それぞれ鋭角エッジとして抽出したエッジ数の全体のエッジ数に占める割合 (r) が 0.05, 0.035, 0.035 で、 max_string_length の値 (sl) が 4, 4, 3、 $min\_feature\_length$ の値 (fl) が 10, 12, 12 となるように設定した結果である。図中で赤く示されているエッジが特徴線として抽出されたエッジで、黄色で示されているエッジはアルゴリズム適用中に検出された、特徴線に隣接するエッジである。

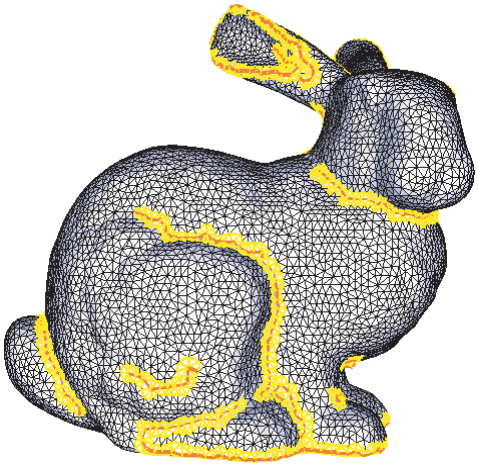
(a) は、最初に鋭角エッジとして抽出したエッジの割合が大きく、また $min\_feature\_length$ の値が小さいため、脚の近くに本来なら特徴線とはいい難い線が抽出されてる。 (b) は、 (a) よりも鋭角エッジの割合を小さくし、 $min\_feature\_length$ の値を大きくした例である。 (a) に比べ、不必要なエッジの数は減少したが、耳の部分や脚の部分が長くつながってしまっている。 (c) は (b) よりも max_string_length の値を小さくした例である。耳や脚の部分での特徴線の抽出がうまくいっている。そこで、ここでは (c) に用いた値を採用し、今後の処理を続けてゆく。

なお、今後の図に登場するウサギのモデルは特に断りのない限り、ここで対象としたものと同じものを扱っている。

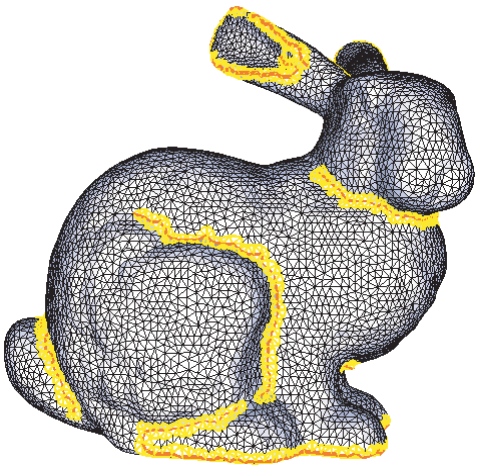
---

expand_feature_curve(halfedge *start*)
vector<halfedge>*detected_feature*
for halfedge h ∈ { *start*, opposite(*start*)}
halfedge $h' \leftarrow h$
do
use depth-first search to find the string S of halfedges
starting with h' and such that:
• two consecutive halfedges of S share a vertex
• the length of $S \leq max\_string\_length$
• $sharpness(S) \leftarrow \sum_{e \in S} sharpness(e)$ is maximum
• no halfedge of S goes backward (relative to h')
• no halfedge of S is tagged as a feature neighbor
$h' \leftarrow$ second item of S
append h' to *detected_feature*
while($sharpness(S) > max\_string\_length \times \tau$)
end // *for*
if (length(*detected_feature*) > *min_feature_length*) **then**
tag the elements of *detected_feature* as features
tag the halfedges in the neighborhood of *detected_feature*
as feature neighbors
end // *if*
end // *expand_feature_curve*

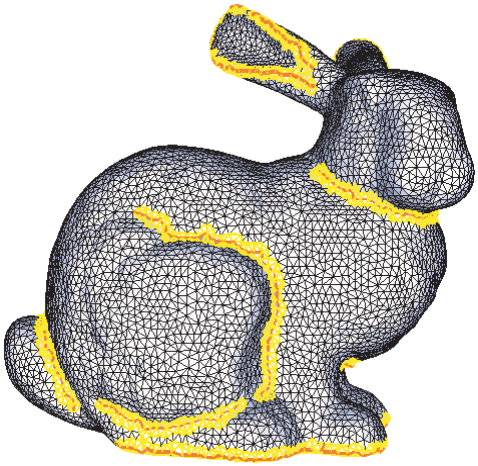
図 4.8: 特徴線抽出のアルゴリズム
(Lévy[49])



(a)



(b)



(c)

図 4.9: 特徴線の抽出結果
(a) $r = 0.05$, $sl = 4$, $fl = 10$　(b) $r = 0.035$, $sl = 4$, $fl = 12$
(c) $r = 0.035$, $sl = 3$, $fl = 12$

パーツ分け

前述の方法で抽出された特徴線を用いて、メッシュモデルのパーツ分けを行う。 Lévy の研究では、得られるパーツのことを**チャート (Chart)** と呼んでいるため、以下では我々の言うところのパーツをチャートと置き換えて記すものとする。

このチャートの生成は、最初に選択した 1 つの面（シード）から徐々に隣接する面へ領域を拡大してゆくことで行う。手法のアルゴリズムは図 4.10に示すとおりであり、処理の概要は次のとおりである。

- 特徴線からの位相的距離 ($distance\_to\_features$) が局所最大となる面をシードに選択する。
- $distance\_to\_features$ が大きいものから順にチャートへ追加し、徐々にチャートを拡大してゆく。
- シードからの距離が閾値 ϵ 以下の距離で異なるチャートが接したらそれらをマージして 1 つのチャートにする。

上記のアルゴリズムを実装し、実験を行ったところ、シードの選択方法とアルゴリズム中に現れるパラメータ ϵ の値に非常にセンシティブで、妥当な結果を得るのが困難であった。そこで、この ϵ の値を小さめに設定してチャートのマージをあまり行わないようにし、以下に述べるアルゴリズムを用いて後からチャートをマージする手法を考案した。

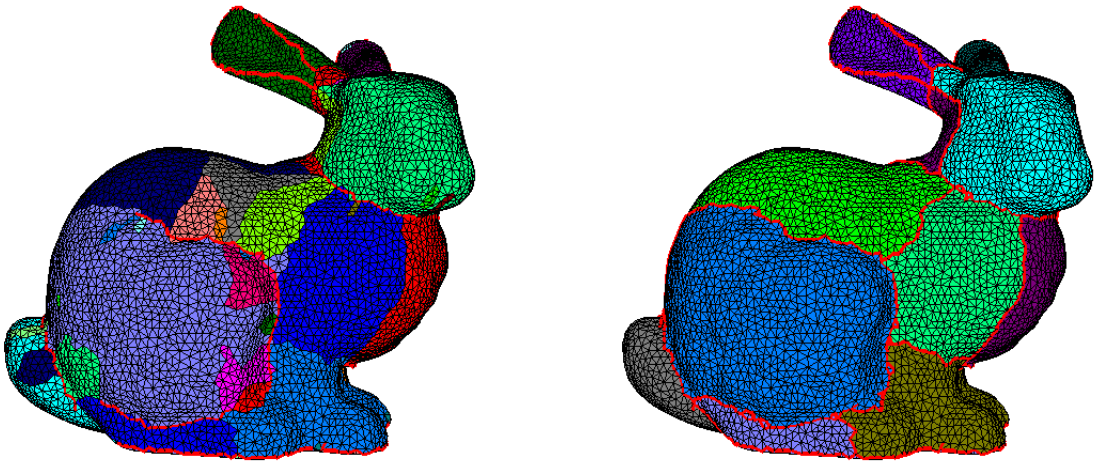
1. 最も小さいチャートを選択する。
2. 選択したチャートに含まれる面の数が閾値 (ex. 全体の 3%) よりも大きければ全体の処理を終える。
3. 他のチャートと接するエッジの数をチャート毎にカウントする。ただしエッジが特徴線に含まれている場合はカウントに含まない。これは特徴線を越えて領域がマージされないようにするためである。
4. 最もカウント数が大きなチャートとマージする。
5. 1. に戻って処理を繰り返す。

このようなアルゴリズムを用いてチャート分けを行った結果を図 4.11に示す。 Lévy の手法において、 ϵ に $distance\_to\_features$ の全体の最大値の 4 分の 1 を用いた結果が (a) である。その後、全体の面の数の 3% よりも少ない面をもつ領域を、新たに提案する手法でマージすることで、 (b) に示す結果を得ることができた。

```
expand_charts
        priority_queue<halfedge> Heap sorted by dist(facet(halfedge))
        set<edge> chart_boundaries initialized with all the edges of the surface
        // InitializeHeap
        foreach facet F where dist(F) is a local maximum
                create a new chart with seed F
                add the halfedges of F to Heap
        end // foreach

        // Charts − growingphase
        while(Heap is not empty)
                halfedge h ← e ∈ Heap such that dist(e) is maximum
                remove h from Heap
                facet F ← facet(h)
                facet F_opp ← the opposite facet of F relative to h
                if ( chart(F_opp) is undefined ) then
                        add F_opp to chart(F)
                        remove E from chart_boundaries
                        remove non-extremal edges from chart_boundaries,
                        // (i.e. edges that do not link two other chart boundary edges)
                        add the halfedges of F_opp belonging to
                                chart_boundaries to Heap
                elseif ( chart(F_opp) ≠ chart(F) and
                        max_dist(chart(F)) - dist(F) < ε and
                        max_dist(chart(F_opp)) - dist(F) < ε ) then
                        merge chart(F) and chart(F_opp)
                end // if
        end // while
end // expand_charts
```

図 4.10: 領域分けアルゴリズム
(Lévy[49])



(a) (b)

図 4.11: 特徴線を利用した領域分け
(a) チャートのマージ前 (b) チャートのマージ後

4.5.2 STRIP 領域の生成

前節の手法で得られた各パーツを、 STRIP の集合で表現するために、パーツに含まれる面をさらに領域分けする。ここで分けられた 1 つの領域が、後に生成する 1 つの STRIP に対応することになる。そのため、ここで分けられる領域は、滑らかな帯状の形をしていることが望ましい。以降では、この領域のことを **STRIP 領域**と呼ぶこととする。

ここでは、各パーツをその輪郭と特徴線からの等距離線によって領域分けすることで細長い領域を生成する手法を提案する。この「距離」には、輪郭と特徴線からの位相的な距離を用いる。なお、ここでの特徴線とは、前節でパーツ分けを行う前処理として抽出した特徴線のことである。パーツに含まれる各面に対してこの距離を設定し、その面の値に基づいて等距離線の生成を行う。この位相的な距離の設定は次のように行う。

1. パーツの輪郭または特徴線に接する面に値 1 を設定する。輪郭からの距離のカウンタ値 $d = 1$ を設定する。
2. まだ距離が設定されていない全ての面について、隣接する面が値 d を持っていたら、その面に $d + 1$ を設定する。全ての面に距離を設定し終わったら処理を終了する。
3. d の値を 1 増やす。
4. 2. に戻って処理を繰り返す。

この結果は、模式的に表した図 4.12のようになる。なお、図中の太線をパーツの輪郭と見なしている。

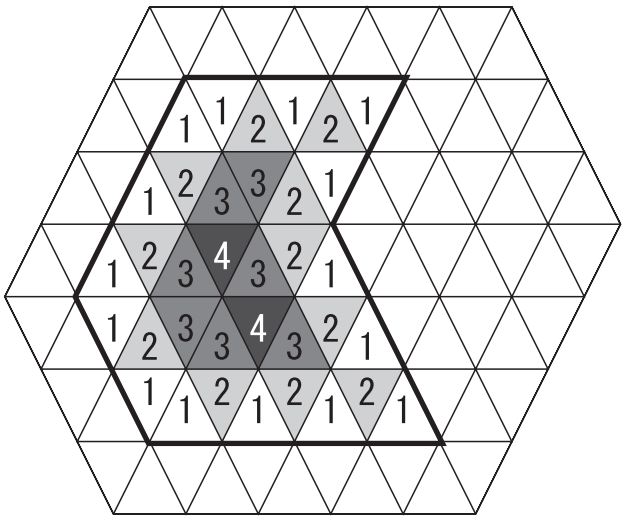
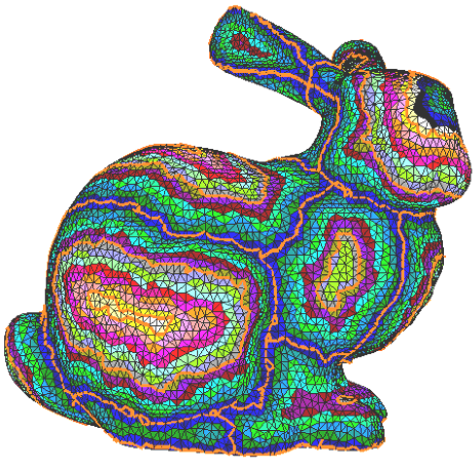


図 4.12: パーツ輪郭からの距離の設定

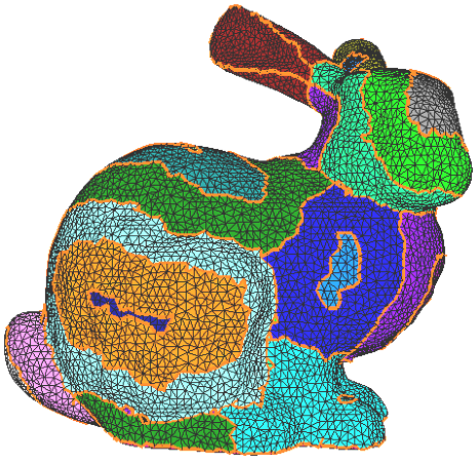
このようにして各面に値を設定した後で、 STRIP 領域の幅に基づいてパーツを複数の STRIP 領域に分ける。 STRIP 領域の幅を h とする場合、面に設定された値が nh と nh+1($n = 1, 2, \ldots$) の間に存在するエッジが STRIP 領域の境界となる。なお、この h の値によって、 STRIP 領域の幅が定まり、値が小さいほど細い STRIP で細かく形状を近似することになる。逆にこの値が大きいと、幅の太い STRIP で形状を近似するため、形状の細かい特徴は失われやすくなる。

このように輪郭と特徴線からの等距離線による分割を行うことで、帯状の輪の形をした STRIP 領域と、一番内側の円盤と同位相の STRIP 領域が得られる。これらの STRIP 領域の中で、特に一番内側の円盤と同位相の STRIP は、非常に小さくなる場合があるので、閾値より小さいもの（領域に含まれる面の数が少ないもの）は隣接する領域とマージ処理を行う。

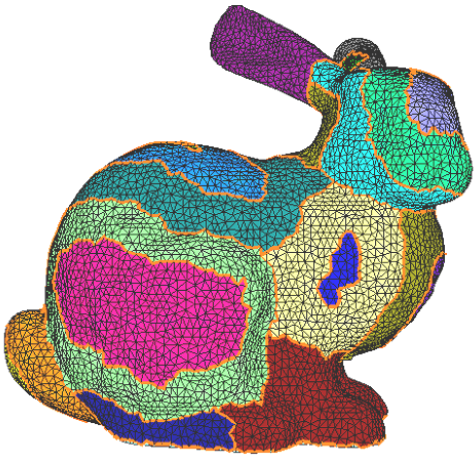
図 4.13は、この手法を用いて STRIP 領域への分割を行った例である。 (a) は各面に設定されたパーツ輪郭からの距離の値に応じて面の色分けを行った例で、オレンジ色の線はパーツの輪郭を表す。 (b) は幅 12 で領域の分割を行った例である。 STRIP 領域ごとに色分けを行っている。 (c) はその後に、閾値として 60 よりも面の数が小さな領域を隣接する領域にマージした結果である。



(a)



(b)



(c)

図 4.13: 特徴線を利用した領域分け
(a) パーツ輪郭からの距離による色分け (b)STRIP 領域の生成
(c) 小さな STRIP 領域のマージ処理

4.5.3　特徴線と中心線の追加

本手法では、特定の辺に含まれる頂点だけを残して、それ以外の頂点を簡略化によって削除することを行う。このような簡略化によって、元のメッシュモデルを、内部に頂点を持たない STRIP の集合にすることができる。このような、簡略化の際に残す辺は展開のときに必ず切断される辺となるため、以降では**切断線**と呼ぶこととする。

ところで、前節の結果として得られた STRIP 領域の境界だけを切断線として簡略化を行った場合、 STRIP 領域内の特徴は消失してしまうという問題がある。これは、具体例として図 4.14を見るとわかりやすい。 (a) の赤い線が STRIP 領域の境界であるとした場合、この赤い線上の頂点だけを残した簡略化を行うと (b) のように、中央の特徴が消失してしまう。

そこで、領域の中心部に新たに切断線を追加することで、元の形状特徴を維持することを行う。図 4.15では、中央の領域の中心部に切断線を追加し、この切断線に含まれる頂点を残すように簡略化を行った例である。メッシュの簡略化を行った後でも元の形状特徴が残されていることがわかる。このような切断線の追加は、パーツ分けの前処理として生成した特徴線のうち、パーツ境界からある一定の距離（例えば STRIP 領域の生成で用いた h の 3 分の 1 など）離れた部分を切断線として追加することで行う。

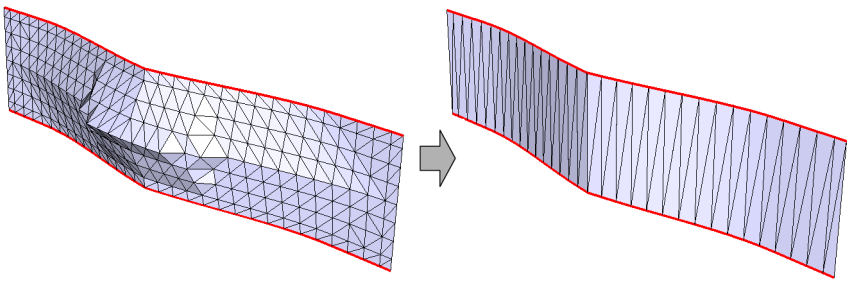


図 4.14: STRIP 領域の境界を残したメッシュ簡略化

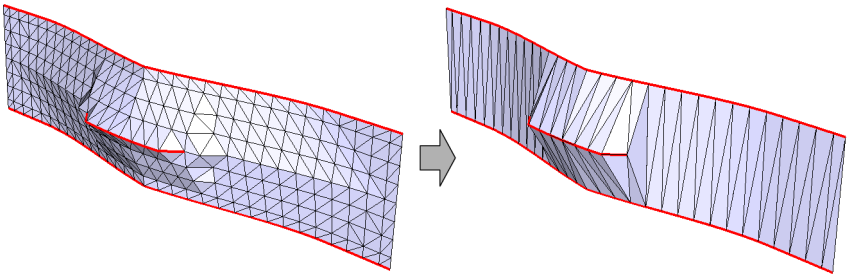
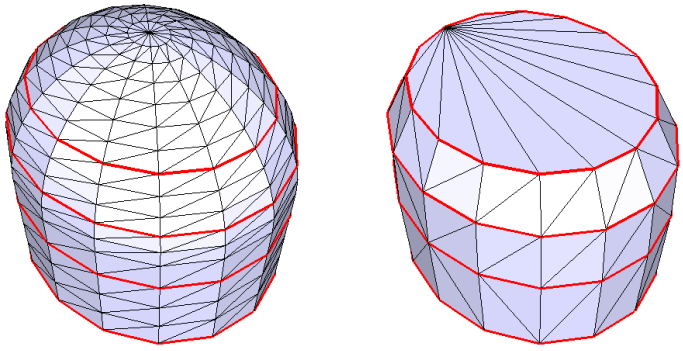


図 4.15: 特徴線を切断線を追加したメッシュの簡略化

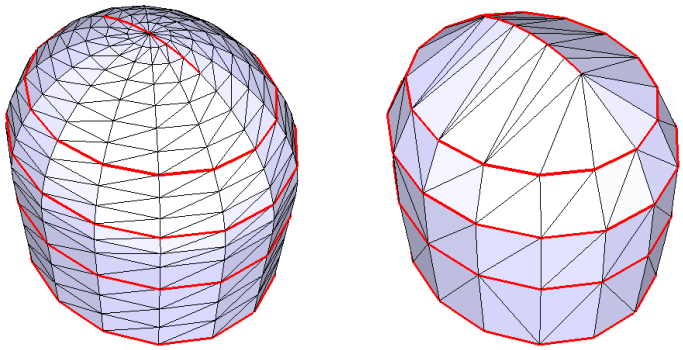
また、丸みを帯びた形状特徴も重要であるが、特徴線が現れない箇所は丸みが無くなってしまう問題がある。これは、具体例として図 4.16を見るとわかりやすい。 (a) の赤い線が STRIP 領域の境界であるとした場合、この赤い線上の頂点だけを残した簡略化を行うと (b) のように、中央の丸みを帯びた特徴が消失してしまう。

そこで、領域の中心部に新たに切断線を追加することで、元の形状特徴を維持することを行う。図 4.17では、中央の領域の中心部に切断線を追加し、この切断線に含まれる頂点を残すように簡略化を行った例である。図 4.16に比べると、メッシュの簡略化を行った後で元の形状特徴がよく残されていることがわかる。このような中心線は、次節で述べる方法で生成する。



(a) (b)

図 4.16: STRIP 領域の境界を残したメッシュ簡略化



(a) (b)

図 4.17: 中心部に切断線を追加したメッシュの簡略化

輪郭ループの縮退による中心線の抽出

ここでは、図 4.17のような中心線を、円盤と同位相のメッシュから自動的に抽出する方法を提案する。

画像認識の分野においては、 2 次元の閉領域から中心線を抽出する手法はよく研究されているが、 3 次元のメッシュを構成するエッジから中心線を抽出する手法は、そのニーズが無いためか該当する研究は見当たらなかった。そこで、領域の輪郭を構成する閉ループを徐々に内部に向かって縮小させることで中心線を抽出する、次のようなアルゴリズムを考案した。

1. 円盤と同位相の領域に含まれる全ての面を面リスト F に追加する
2. F の輪郭を成す反時計回りのエッジのループ L を作成する
3. F に含まれるループ L に接する面のうち、最も領域境界からの位相的距離が小さいものを F から削除し、 L を更新する。ただし削除する面（三角形）とループ L の接し方に応じて、ループ L の更新を次のようにする。なお、参考とする図 4.18〜 4.20において、 F に含まれる面を薄いグレー、削除対象となっている面を濃いグレー、 L を太い実線で示す。また矢印は L の向きを示し、色の同じ矢印は L 内で連続しているが、色の異なる矢印同士は L 内で連続していないことを示す。

   面の 1 辺が L に含まれる場合 L に含まれる 1 辺を削除して、残りの 2 つの辺を L に追加する（図 4.18）。

   面の 2 辺が L に含まれる場合 図 4.19(a) のような場合は L に含まれる 2 辺を削除して残りの 1 つの辺を L に追加する。図 4.19(b) のような場合は、どちらか一方を削除して、三角形に含まれるそれ以外の 2 つの辺を L に追加する。

   面の 3 辺が L に含まれる場合 図 4.20(a)、 (b) のような場合は連続して L に含まれる 2 辺からどちらか 1 辺を削除して、残りの 2 辺を L に追加する。図 4.20(c) のような場合は、 3 辺からいずれか 1 辺を削除して、三角形に含まれるそれ以外の 2 つの辺を L に追加する。
4. F が空になるまで 3. を繰り返す。

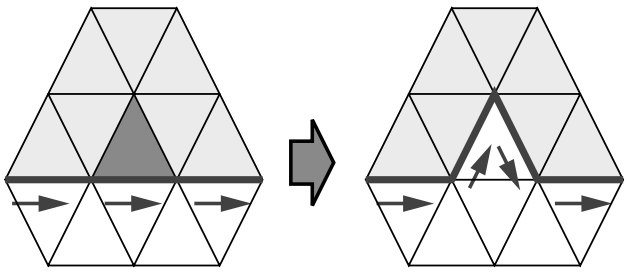


図 4.18: 輪郭ループの更新（1 辺が接する場合）

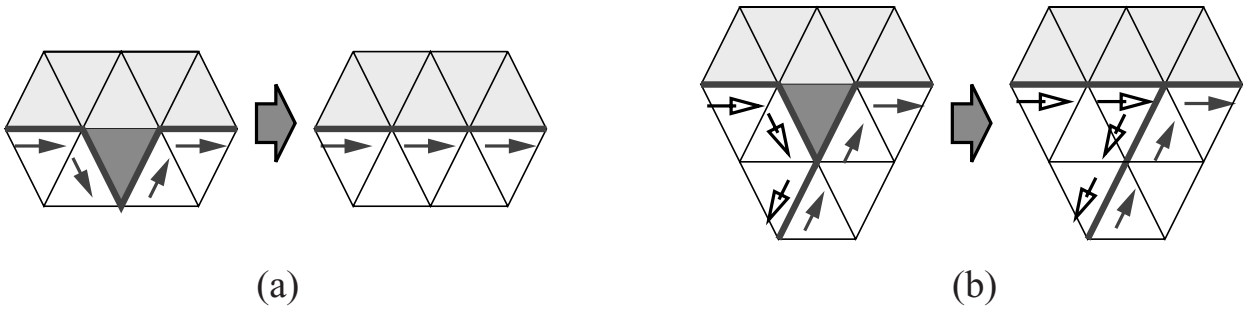


図 4.19: 輪郭ループの更新（2 辺が接する場合）

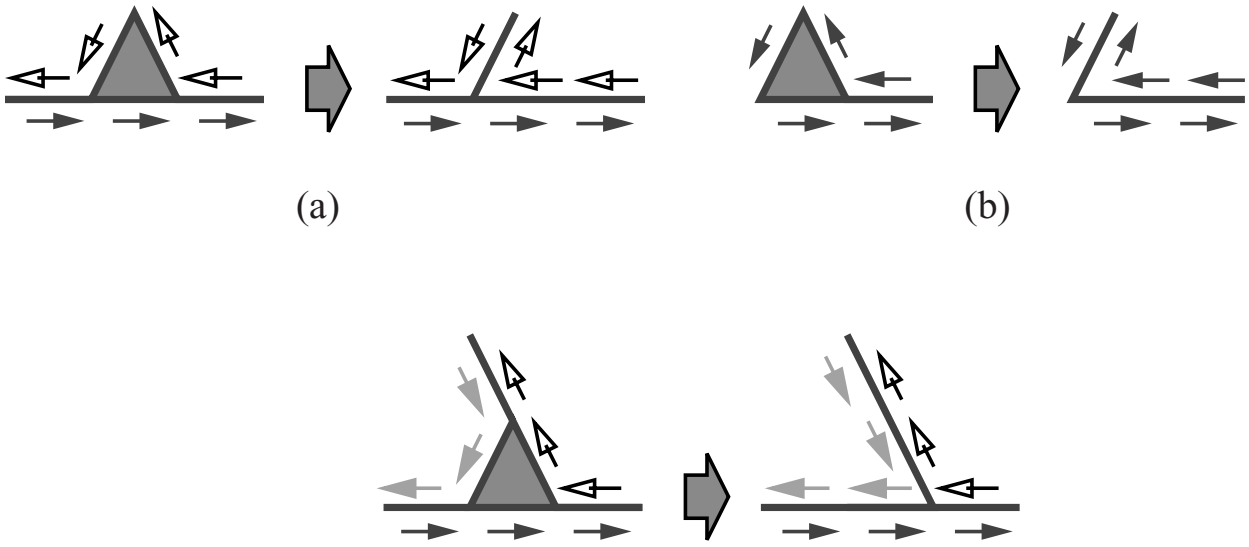


図 4.20: 輪郭ループの更新（3 辺が接する場合）

中心線の簡略化

前述のアルゴリズムを適用することで、最終的には内部に面を含まない縮退したループを得ることができる。このループは領域の輪郭から内側に向かって縮退したものであるから、元のメッシュ領域の中心線を成すと見なすことができる。

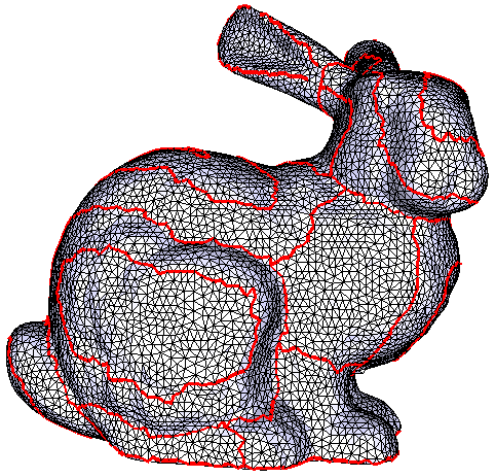
前節の結果として得られた STRIP 領域（図 4.21(a)、図中の赤い辺が領域の境界）の中で、特に円盤と同位相の領域に対して、この手法を適用した結果は図 4.21(b) のようになった。領域の輪郭が中央に縮退して、複雑な形をしていることが確認できる。このような中心線は意図する中心線を含んではいるが、切断線としては複雑すぎるため工作には適切でない。

そこで、さらにこの中心線を次のようにして簡略化することとした。

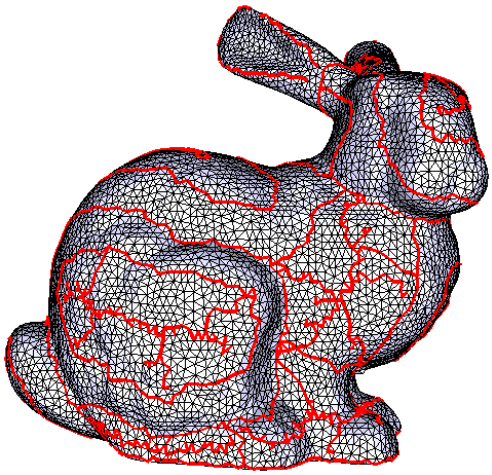
1. 得られた中心線に含まれるエッジを枝、頂点を節点とするグラフを作成し、次の処理を行う。
   (a) グラフの葉に当たる頂点をリスト L に追加する。
   (b) L に含まれる頂点を 1 つずつ削除する。最初に含まれた頂点の数に対する一定の割合 (*delete_vertex_percent*) の頂点が削除されたら全体の処理を終了する。
   (c) リストが空になったら、 (a) のステップへ戻る。
2. 領域の境界からの距離が一定値 (*mindist_from_outerloop*) 以上近いものも削除する。

ここでは、 *delete_vertex_percent* の値に 85% を使用し、 *mindist_from_outerloop* の値を 4 にした。このようにして得られた結果が図 4.21(c) である。円盤と同位相の閉領域について、妥当な中心線が生成されたことを確認できる。

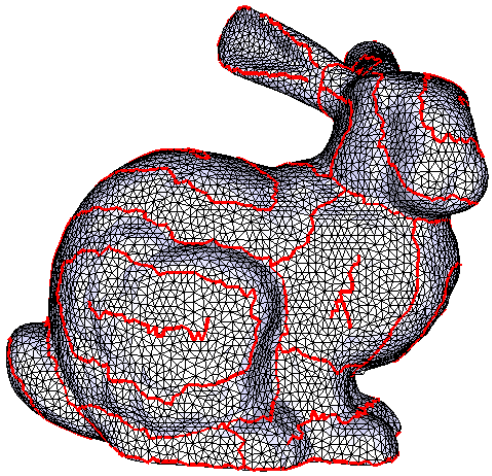
なお、ここで生成された中心線は図 4.17の例のように、 STRIP 領域の境界線と同様に簡略化するときにも保持され、工作するときの切断線となる。



(a)



(b)



(c)

図 4.21: 中心線の生成
(a)STRIP 領域の境界と特徴線 (b) 円盤と同位相の STRIP 領域での中心線の生成 (c) 中心線の簡略化

4.5.4 境界の平滑化

前節までの処理によって、 STRIP の集合でモデルを近似した時に切断辺となる辺の抽出を行えた。この切断辺は、 STRIP 領域の境界と前節で生成した中心線から成る。

ところで工作の時には、この辺に対して切り取りと貼り合わせを行うことになるため、これらの辺の連続から成る切断線は、なるべく滑らかであった方がよい。これは 3.3節のコスト評価式において、 stoit 点の数が少ないほうが工作のコストが小さくなることからもわかる。

そこで、前節までの手法で得られた切断線の平滑化を行う。ここでは、メッシュの形状は変化させずに切断する辺を変更することで平滑化を行う位相的に平滑にする手法と、メッシュの形状（メッシュに含まれる頂点の座標値）を変更して切断する辺を幾何的に平滑にする手法の 2 通りの方法を両方行うものとした。

位相的な平滑化

ここで述べる位相的な平滑化の内容を図 4.22に示す。図 4.22左側の灰色の三角形ように、 1 つの三角形に 2 つの連続する切断線が存在した場合、この 2 つの切断線を残りの 1 つの辺に置き換えることで切断線の平滑化を行う。なお、この処理は対象となる切断線が他の切断線との交差をしていない場合にのみ適用する。

この手法を前節の手法で得られた切断線に適用した結果は図 4.23のようになった。 (a) が適用前で、 (b) が適用後である。

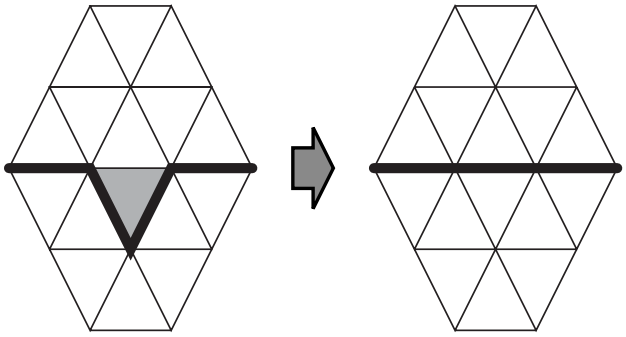
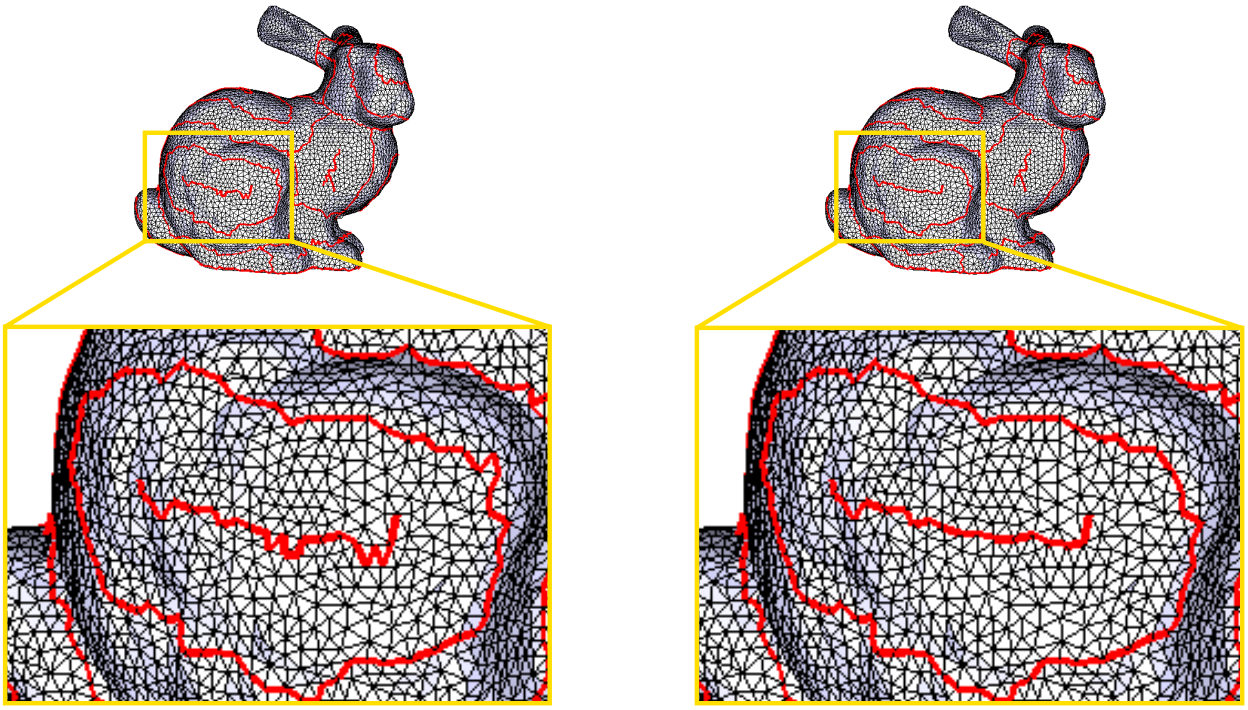


図 4.22: 切断線の位相的な平滑化



(a) (b)

図 4.23: 切断線の位相的な平滑化を施した例
(a) 位相的な平滑化処理の前 (b) 位相的な平滑化の適用後

幾何的な平滑化

前述までの手法では、元のメッシュ形状は完全に保存されていたが、ここでは、切断線に含まれる頂点の位置を動かすことで、切断線の平滑化を行う。この平滑化の手法には、式 4.1で表されるような平滑化のラプラシアンオペレータ [63] を用いた。

$$p_{n+1}[i] = \frac{1}{4}p_n[i-1] + \frac{1}{2}p_n[i] + \frac{1}{4}p_n[i+1] \tag{4.1}$$

上式の $p[i]$ は切断線上の点の座標値を表し、 $p[i-1], p[i+1]$ はその点に切断線上で隣接する 2 つの点の座標値を表す（図 4.24）。この処理を座標値が収束するまで繰り返すと、曲線は直線に収束するが、適度な回数施すことで、微小な凹凸が軽減された滑らかな曲線を得られることが知られている。

ここでは、切断線に含まれる頂点のうち、その頂点で切断線が T 字交差を成さないものについて、上記のラプラシアンオペレータを施す。収束するまで処理を繰り返すと、形が縮退し、元の頂点の位置から大きく移動してしまうことがあるので、ここでは 2 回適用することとした。前述の位相的な平滑化処理の後に、この処理を施した結果を図 4.25に示す。

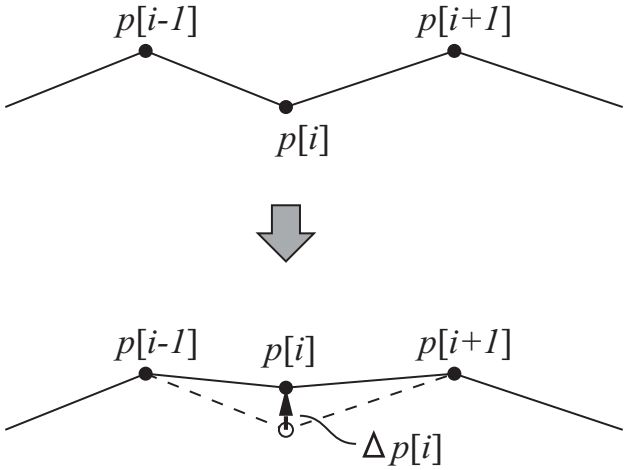
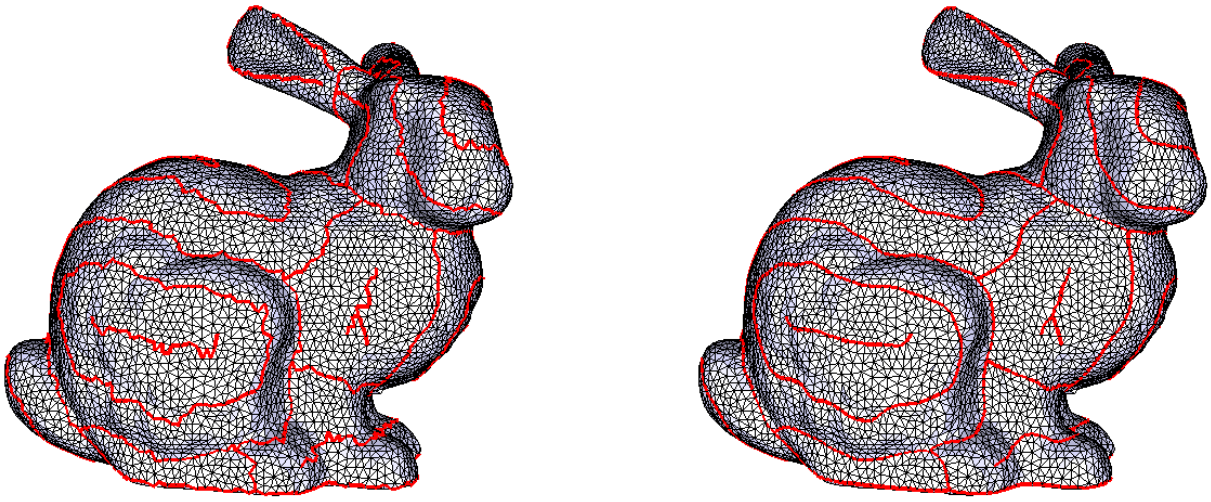


図 4.24: 隣接頂点の位置に基づく平滑化



(a) (b)

図 4.25: 切断線の幾何的な平滑化
(a) 頂点移動による平滑化処理の前 (b) 頂点移動による平滑化の適用後

4.5.5　メッシュ簡略化

前節までの処理で、 STRIP 領域とその輪郭となる切断線が決定した。この STRIP 領域の内部に含まれる頂点を除去し、切断線（輪郭線および 4.5.3節で追加した中心線）上の頂点のみを残したメッシュの簡略化を行うことで、 STRIP を作成する。

メッシュの簡略化については 4.3節でまとめたように、様々な手法が研究されてきたが、ここでは、近似精度が高く高速で処理ができるものとして知られている Garland ら [54] の手法を採用した。この手法では、 Edge collapse を繰り返し行うことで面の数を減らしてゆく。この手法では、 Edge collapse を行った後で頂点の位置を移動させることを行うが、本手法では切断線上にある頂点は位置の移動を行わないものとする。また、 Edge collapse は切断線上にある稜線には適用しないものとする。さらに、位相的な縮退が発生する場合や、面の反転が起こる場合には、この Edge collapse は行わないものとする。

ある程度処理を繰り返すと、これ以上 Edge collapse を行えなくなる場合がある。その場合には、 Vertex decimation[64] の処理を用いて、頂点の削除と穴埋めの処理を行う。

この処理を行って得られた結果は図 4.26のようになった。この簡略化後の面の数は 3,230 である。

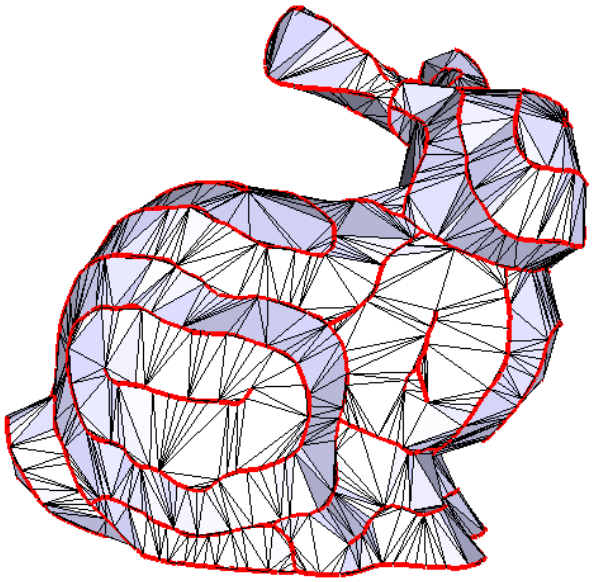


図 4.26: 切断線を残した簡略化

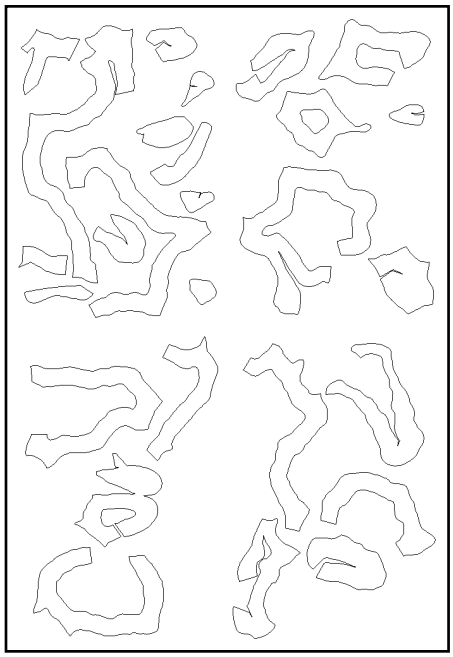
4.5.6 展開図の作成と組み立て

前節で生成された、 STRIP の集合で表現されたメッシュモデルを前章で述べたポリゴンモデルの展開図作成手法で展開する。展開の際には STRIP の境界が切断される辺となるようにすればよく、それ以外には特に気をつけるべき点はない。

この結果として得られた展開図は図 4.27(a) のようになった。 (b) は、元のメッシュが STRIP の集合に近似された様子がわかりやすいように、切断部分を強調表示したものである。これを実際に組み立てた結果が (d) であり、 (c) は STRIP を貼りあわせて各パーツの形を作成した時点のものである。

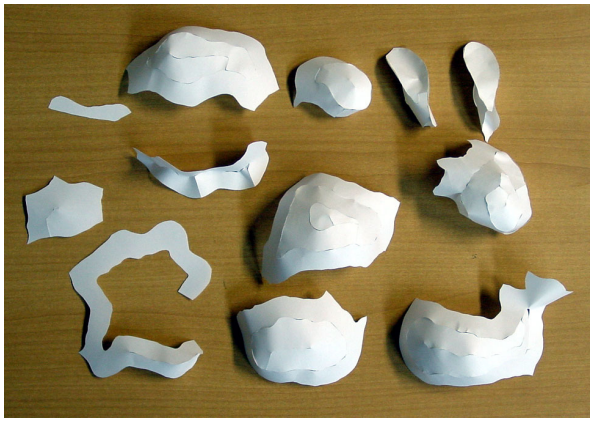
なお、展開図の切り抜きはカッティングプロッターを用いて行ったため、実際の組み立ては STRIP 同士の貼り合わせがメインの作業となった。この作業にはおよそ 3 時間半程度を要した。展開図だけを見ると、どの STRIP とどの STRIP を貼りあわせればよいかの把握が困難であるが、実際の工作の時には、 3.5.3節で提案した工作支援機能を活用したため、展開図と立体の対応関係を容易に把握することができ、大きな問題はなかった。

組み立て後の模型は、紙の柔軟性を活かす事で元のメッシュモデルにみられる滑らかな形状特徴を表現できている。動物などにみられる滑らかな形を紙模型で表現する際には、ここで提案した手法が有効であることが確認できる。



(a)

(b)



(c)

(d)

図 4.27: 展開図の作成と組み立て
(a) 展開図 (b) 切断部分を強調表示したもの
(c) 組み立てた紙模型（パーツ） (d) 組み立てた紙模型

4.6 手法の適用結果

本章で提案した手法は、図 4.9に示すウサギのモデルを中心に実験を行い、新しく考案してきたものであるが、基本的な考え方はハーフエッジ構造で表現可能な任意のメッシュモデルに適用可能である。

そこで、別の例題として、面の数がそれぞれ 18,496 のサイのモデル (A)、 14,176 の恐竜のモデル (B)、 11,996 の人のモデル (C) に本手法を適用して実験を行った。

その結果を図 4.28〜図 4.34に示す。各図の説明と、それぞれのモデルに使用した本手法に必要となるパラメータの値は次の通りである。なお、各パラメータの値は、処理の結果を見ながら適当と思われる値を試行錯誤で求めた。

(a) 特徴線の抽出を行った結果の図である。鋭角エッジとして抽出したエッジ数の全体のエッジ数に占める割合 (r) にそれぞれ (A)0.06、 (B)0.07、 (C)0.1、 max_string_length の値 (sl) にそれぞれ 3、 $min\_feature\_length$ の値 (fl) にそれぞれ (A)12、 (B)8、 (C)9 を用いた。

(b) パーツ分けを行った結果の図である。隣接する領域のマージ判定に用いる ϵ の値に、

$distance\_to\_features$ の全体の最大値の (A)(B)$\frac{1}{5}$、 (C)$\frac{1}{4}$ を用い、その後の小さい領域をマージするための閾値には全体の面の数の (A)2%、 (B)(C)5% を指定した。

(c) STRIP 領域への分割を行った結果の図である。 STRIP 領域の位相的な幅には (A)11、 (B)(C)10 を指定した。また、小さい STRIP をマージするための閾値には、面の数 (A)250、 (B)(C)80 を用いた。

(d) 円盤と同位相の STRIP 領域へ中心線を追加した図である。輪郭を縮退させて得られた中心線から 85% 分の枝刈りを行い、さらに輪郭からの位相距離が (A)(C)6、 (B)4 以下の中心線の削除を行った。

(e) 切断線を平滑化した図である。位相的な平滑化ののち、ラプラシアンオペレータを 2 回施して頂点の位置を動かした結果である。

(f) 切断線の頂点のみを残して面の簡略化を行った図である。

(g) STRIP の集合によって形状が構成されていることがわかりやすいように、切断線箇所を強調表示した図である。

(h) 得られた STRIP の集合を平面に展開した結果である。

(i) 実際に組み立てを行った結果である（A のみ）。ウサギの例題と同じく、カッティングプロッタで展開図の切抜きを行った。組み立てに要した時間は 3 時間程度であった。

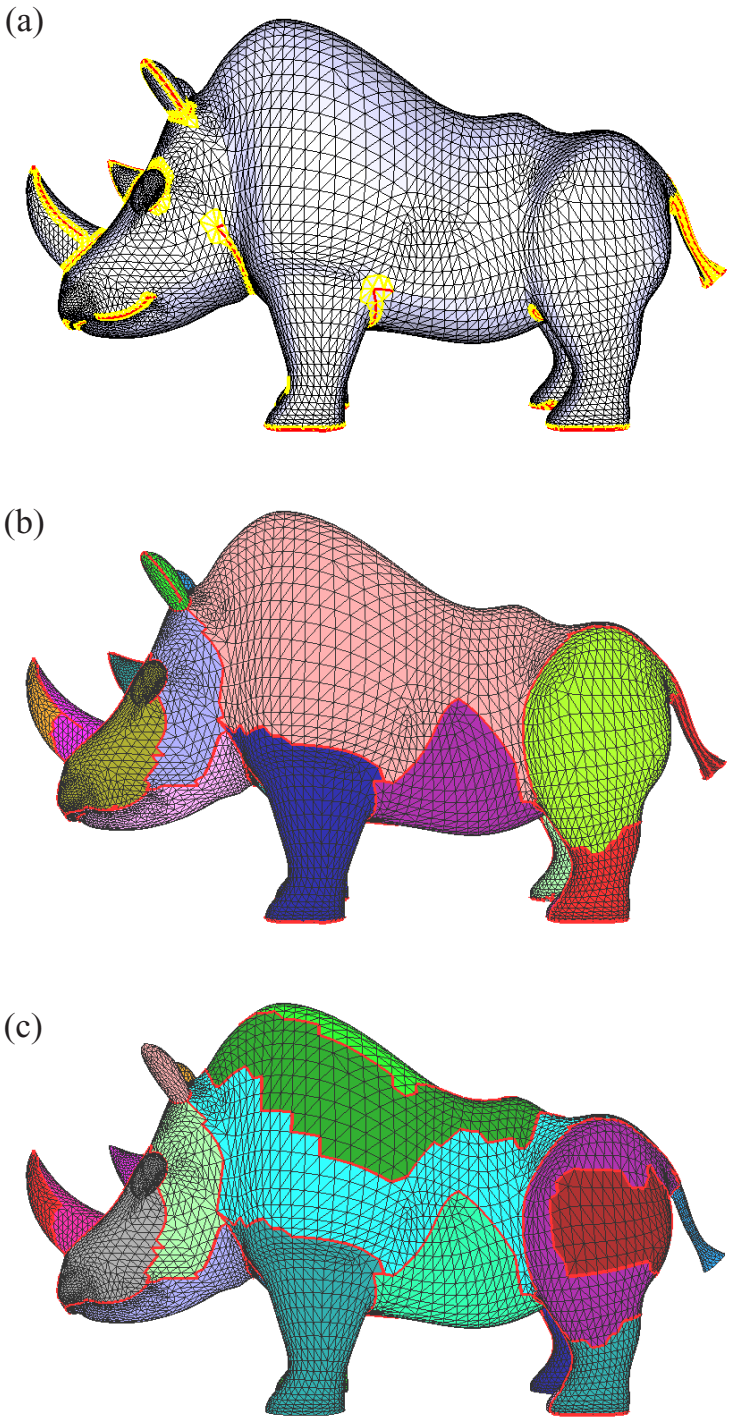
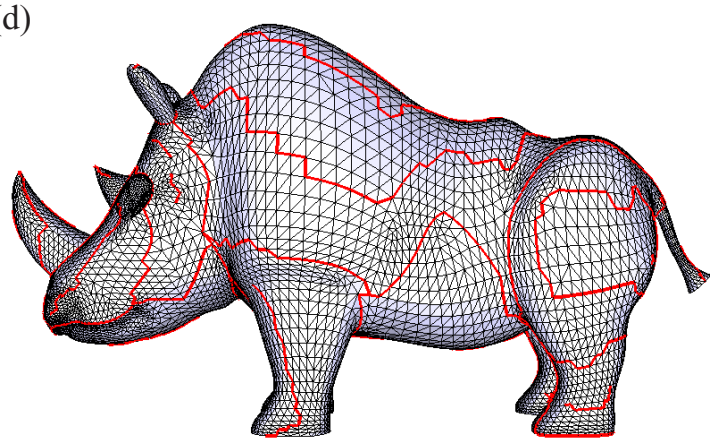
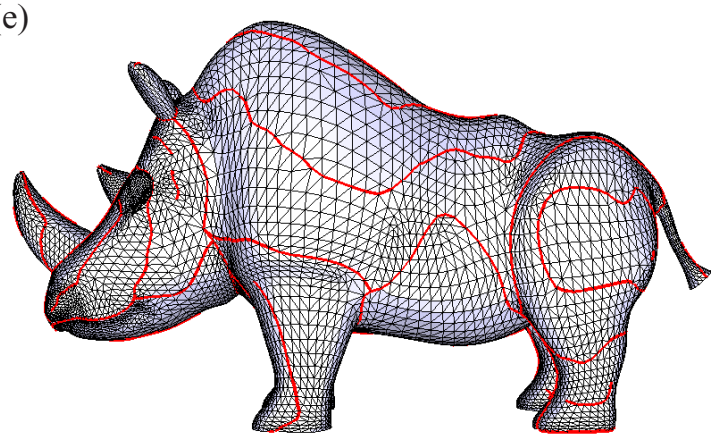


図 4.28: サイのモデルへの適用結果 1
(a) 特徴線の抽出 (b) パーツ分け (c) STRIP 領域の生成

(d)



(e)



(f)

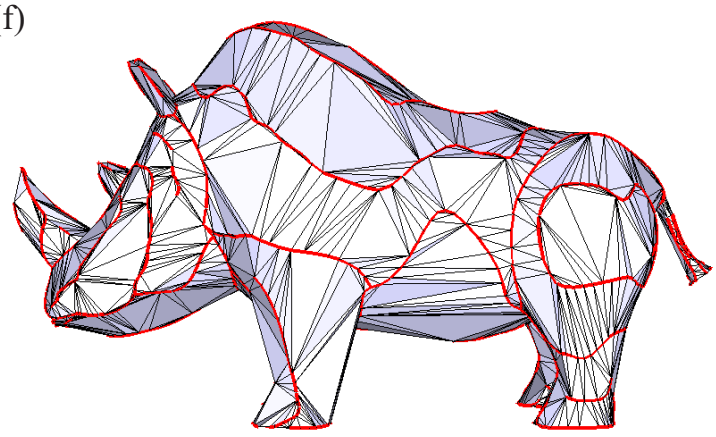
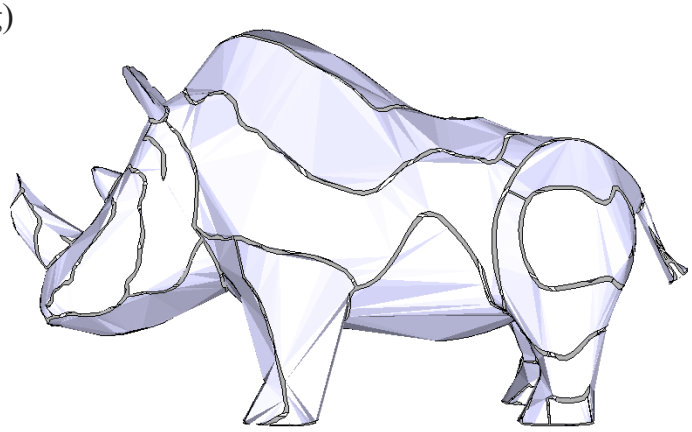
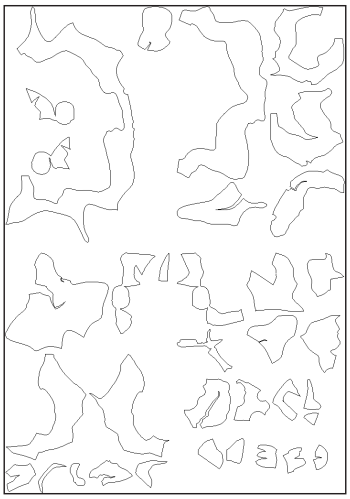


図 4.29: サイのモデルへの適用結果 2
(d) 中心線の生成 (e) 切断線の平滑化 (f) メッシュ簡略化

(g)

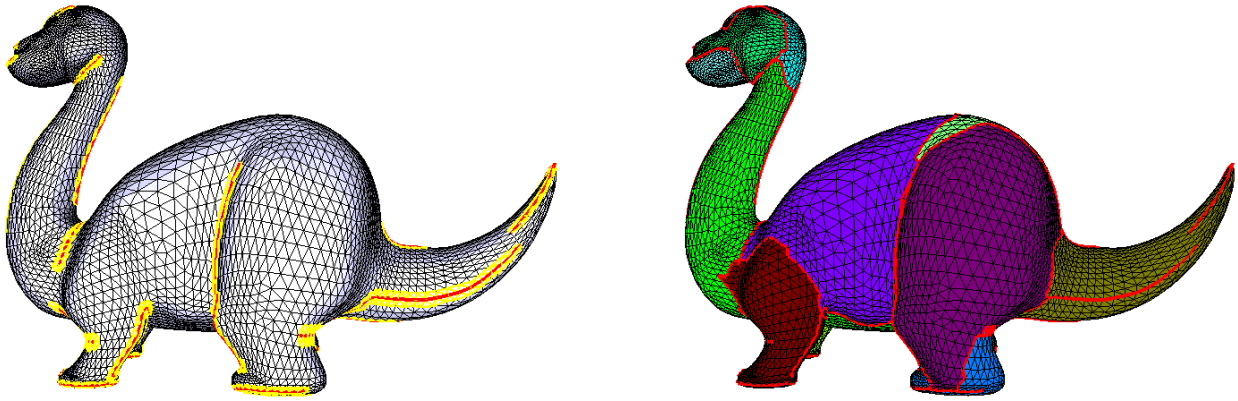


(h)



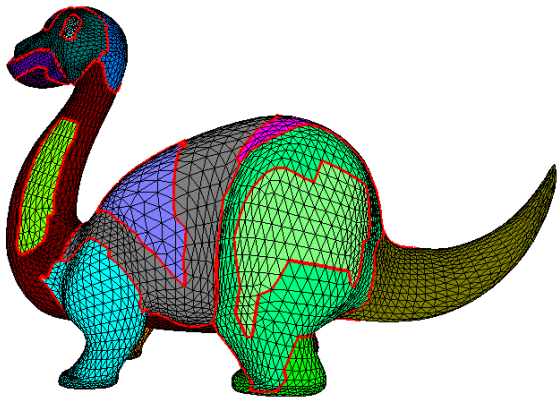
(i)

図 4.30: サイのモデルへの適用結果 3
(g) 切断線の強調表示 (h) 展開図 (i) 作成した紙模型

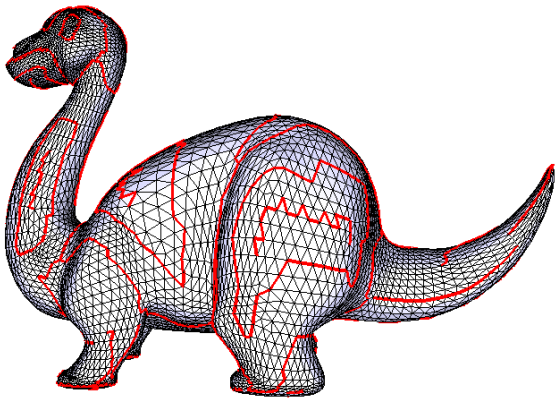


(a)

(b)

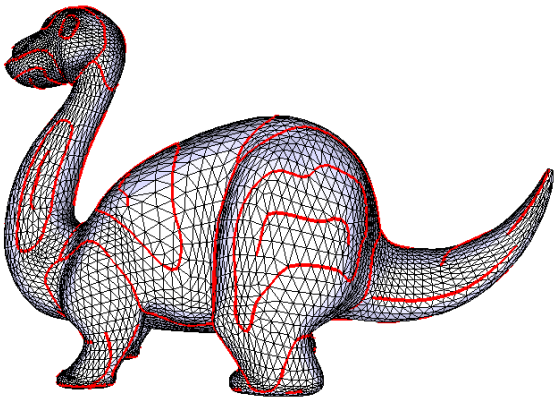


(c)

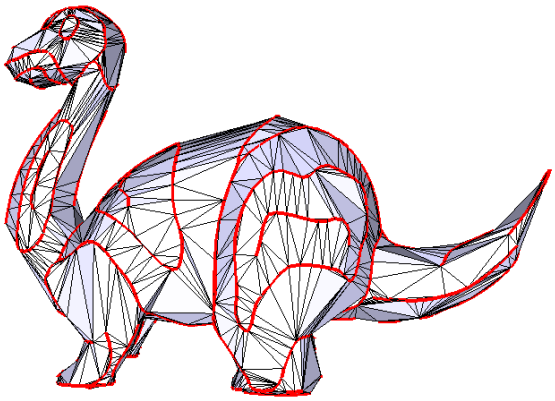


(d)

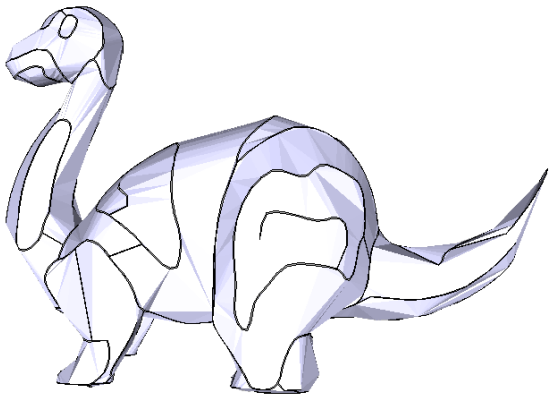
図 4.31: 恐竜のモデルへの適用結果 1
(a) 特徴線の抽出 (b) パーツ分け (c) STRIP 領域の生成 (d) 中心線の生成



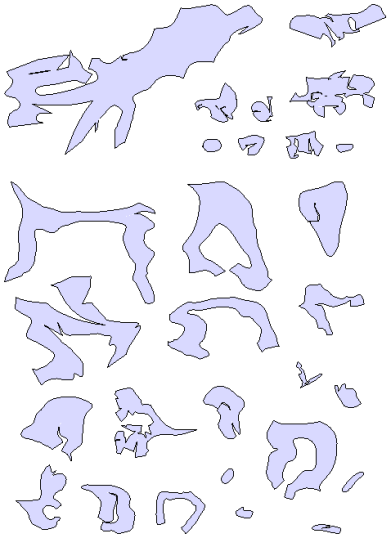
(e)



(f)



(g)



(h)

図 4.32: 恐竜のモデルへの適用結果 2
(e) 切断線の平滑化 (f) メッシュ簡略化 (g) 切断線の強調表示 (h) 展開図

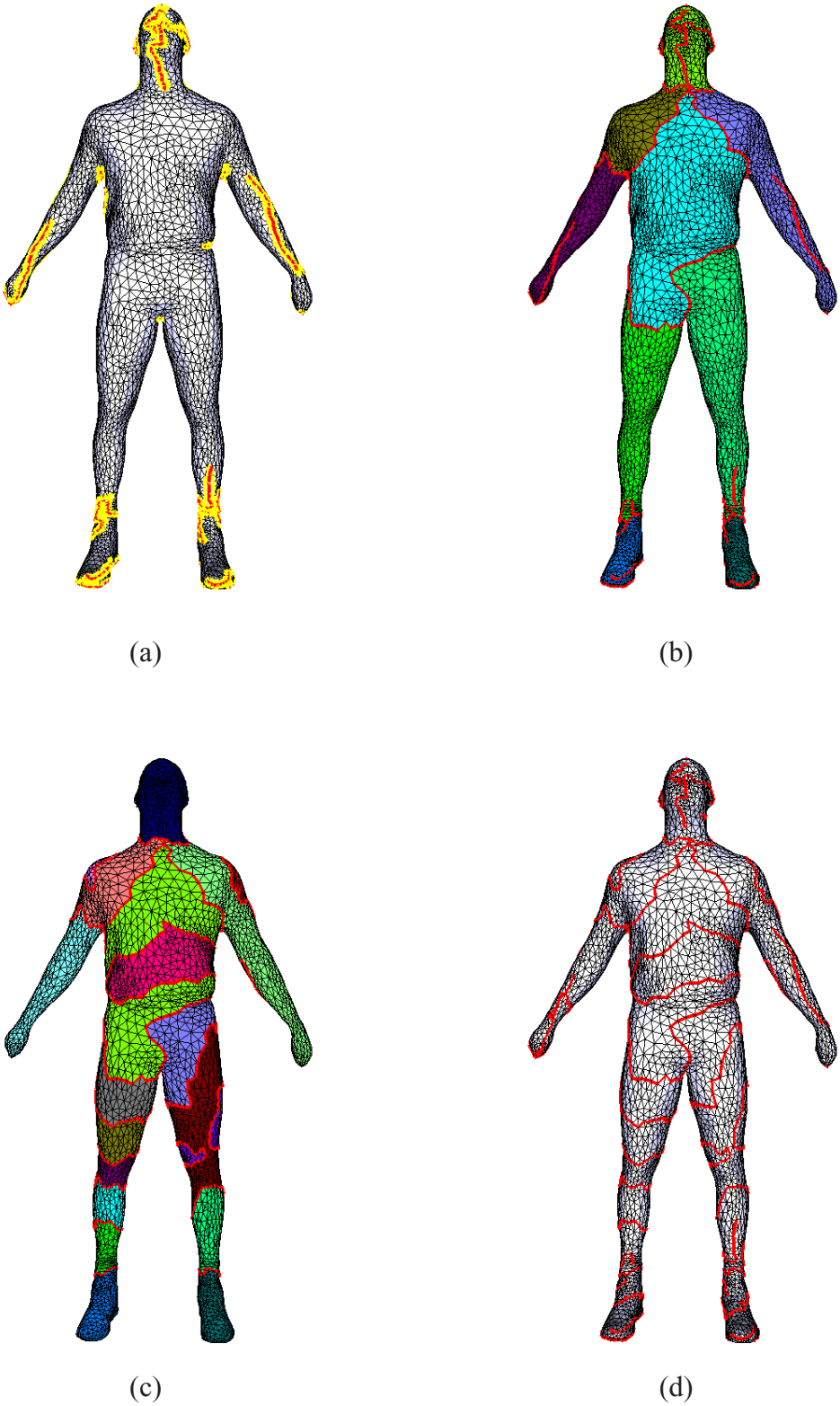
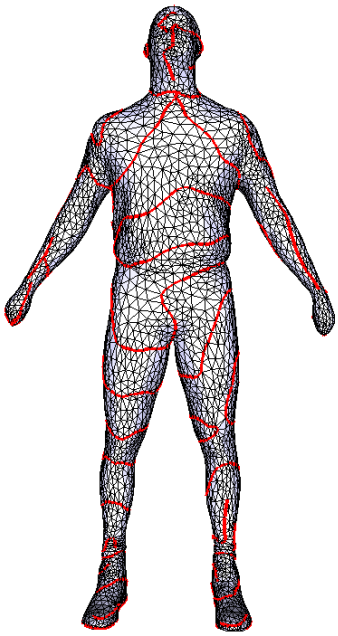
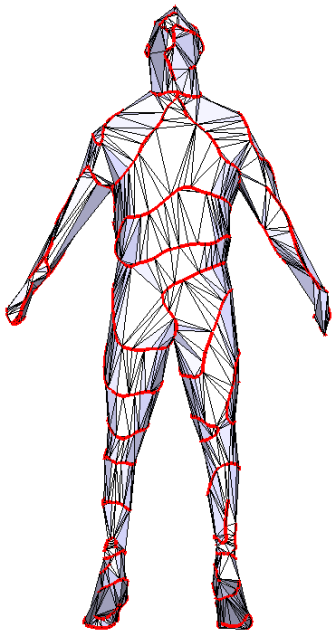


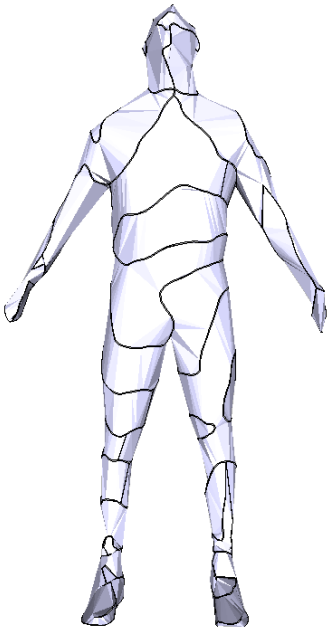
図 4.33: 人のモデルへの適用結果 1
(a) 特徴線の抽出 (b) パーツ分け (c) STRIP 領域の生成 (d) 中心線の生成



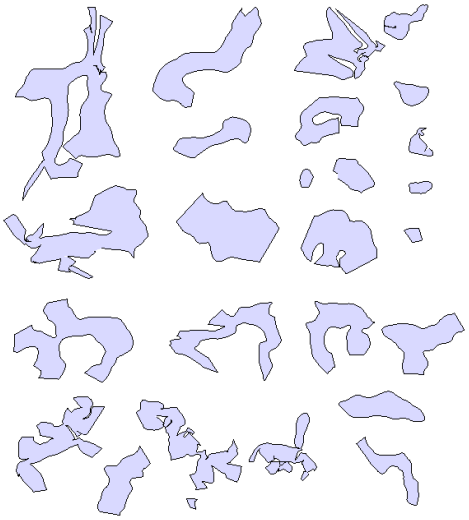
(e)



(f)



(g)



(h)

図 4.34: 人のモデルへの適用結果 2
(e) 切断線の平滑化 (f) メッシュ簡略化 (g) 切断線の強調表示 (h) 展開図

4.7 組み立てコストの評価

前節までで、メッシュモデルを STRIP の集合で近似する手法を提案し、実際に得られた展開図を組み立てることで近似的な紙模型を作成できることが確認できた。この紙模型は紙の柔軟性を活かした滑らかな形状特徴を持ち、また簡略化後の面の数が数千を超えているにもかかわらず、折り曲げ操作を省くことで、現実的な時間で組み立てることができた。

本節では、3.3節でまとめた組み立てコスト評価式を用いることで、この手法によって得られた展開図と、単純なメッシュ簡略化を用いて得られた展開図のコストの比較を行う。

3.3節でまとめた展開図の組み立てコスト評価式は次の通りである。

$$\begin{aligned} C_{total} &= w_{cut\_straight}L_{cut\_straight} + w_{cut\_curve}L_{cut\_curve} + w_{cut\_stoit}N_{cut\_stoit} \\ &\quad + w_{halfcut}L_{break} + (w_{break} + w_{break\_stoit})N_{break} \\ &\quad + w_{paste}N_{paste} \end{aligned} \tag{4.2}$$

上記の評価式は、ポリゴンモデルの展開図を評価するための式であったが、本手法で作成した STRIP の集合によって表現されるモデルに対しても適用することができる。ただし、本手法では紙の柔軟性を用いて曲面を表現していることと、細長い三角形の集合で各 STRIP が構成されていることから、次のような評価の方法を行うこととした。

- 各 STRIP の輪郭は長さの短いエッジの集合で近似表現された曲線なので、切り抜きは全てフリーハンドで行うものとし、切抜きのコストには $w_{cut\_curve}L_{cut\_curve}$ を用いる。L_{cut_curve} は、展開図の輪郭線の総和である。また、フリーハンドで行うので stoit 点は評価に含めないこととする。

- 本研究のモデルは、紙の柔軟性を用いて曲面を表現するため、山折り、谷折りという折り曲げの概念が存在しない。そこで、折り曲げ線に起因するコストはゼロとする。具体的には、$w_{halfcut}, w_{break}, w_{break\_stoit}$ の値をゼロとする。

- 貼り合わせは、貼り合わせを行う連続するエッジについて一定間隔でセロハンテープを用いて行うものとし、「貼り合わせ回数」は「貼り合わせを行う辺の総延長」を「貼り合わせ間隔」で割った値とする。具体的には、1.5cm 間隔でセロハンテープで貼り合わせるものとし、$N_{paste} = L_{paste}/1.5$ とする。なお、対象とするモデルが二多様体（全てのエッジが 2 つの面に属する）場合、$L_{paste} = L_{cut}/2$ である。

上記の内容を踏まえて式 4.2の評価式をまとめなおすと、本手法で作成した STRIP 集合による近似モデルの組み立てコスト評価式は、次のような簡単な式にすることができる。

$$C_{total} = w_{cut\_curve}L_{cut\_curve} + \frac{w_{paste}L_{paste}}{1.5} \tag{4.3}$$

式 4.3の係数 w_{cut_curve} と w_{paste} は、3.3節で求めた表 3.5の計測値から、それぞれ $w_{cut\_curve} = 1.1, w_{paste} = 7.7$ である。本手法の例題としたウサギとサイのモデルの幾何的な値と、この係数から算出した組み立てコストは表 4.1のようになった。組み立てコストは「秒」を単位として算出できることから、この値だけを参照すると、共に約 50 分で組み立てらることになる。しかし、カッティングプロッタを用いても、どちらも 3 時間程度かかったことを考慮すると、

およそ 3 倍程度の誤差があることになる。これは、事前にセロハンテープを一定の長さに切り取って準備しておくことをしていなかったことによる工作の方法の問題点とともに、複雑なモデルほど組み立て時の思考に要する時間が増すことや、細かい部分の組み立ての困難さに依存する誤差だと思われる。このように、算出された値自体は、そのまま時間の見積もりには直結しないが、工作のコストの相対評価には有効に使用できると思われる。

表 4.1: 係数の算出結果

	L_{cut_curve}	L_{paste}	組み立てコスト
ウサギ	802	401	2942
サイ	871	435	3193

この結果と比較するために、 QSlim[55] を用いて、それぞれの元のモデルを簡略化したものを、 3.4節で述べた「エッジ長とエッジ角」の評価を行って展開したもののコストを表 4.2と表 4.3にまとめる。なお、これは全てのエッジを折るものとして算出している。この結果、本手法で提案した STRIP の集合による近似展開図の作成コストは、ほぼ面の数を 250 〜 300 程度まで従来の手法で簡略化したものと同じ程度であることがわかった。これは、実際の経験からも、ほぼ正しい評価であると言える。

面の数を 250 〜 300 程度まで減らしたモデルは、表中の図から見て取れるように、形状の大まかな特徴は残っているものの、非常に面が粗く、面が角ばっているので、元の滑らかな特徴は消失してしまっている。これと比較すると、本章で提案した手法は、同程度の組み立てコストで、滑らかな特徴を残した紙模型を作成できることがわかる。

表 4.2: 簡略化されたモデルの面の数と組み立てコスト（ウサギ）

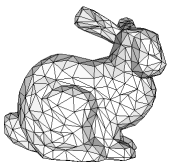
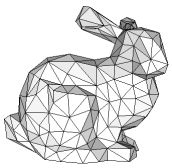
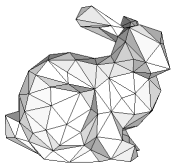
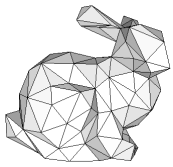
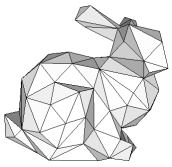
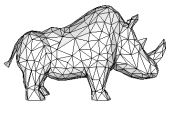
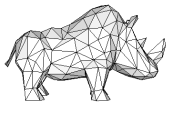
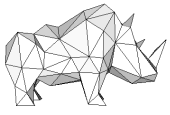
					
面数	1000	500	300	250	200
コスト	11478	5874	3460	2901	2352

表 4.3: 簡略化されたモデルの面の数と組み立てコスト（サイ）

					
面数	1000	500	300	250	200
コスト	11478	5777	3470	2869	2299

4.8 STRIP の平滑化

本章で提案する手法では、 STRIP 領域のメッシュを簡略化することで、内部に頂点を持たない STRIP の集合を生成する。 4.5.5節で述べたメッシュ簡略化では、 Garland ら [54] の手法を用いてメッシュを簡略化することで、 STRIP の生成を行った。 Garland ら [54] の手法には、簡略化後の形状と元のメッシュの形状の誤差が小さく抑えられる特徴がある。しかしその一方で、本手法では切断線上の辺を固定するために極端に細長い扁平な三角形が生成されやすいという問題がある。図 4.26では STRIP 間の接続が失われているように見えるが、位相的な接続は失われていないため、図では見えないほどの細長い三角形が間に存在していることがわかる。

これらの扁平な三角形のエッジは、折れ角が小さいことが多いが、 STRIP の境界付近に集中していて、かつ面積が極端に小さいため、工作の時にはこれらを無視して工作を進めることができる。しかし、本手法の基本的な考え方である「滑らかな STRIP の集合で近似する」ことを達成するためには、この折れ角の小さい辺はなるべく含まれない方がよい。そこで、4.5.5節の簡略化によって得られた STRIP に対して、その内部に含まれる各エッジ e に Edge Swap[65] の手法（図 4.35）を適用することで、より滑らかな STRIP の生成を行うことを試みた。

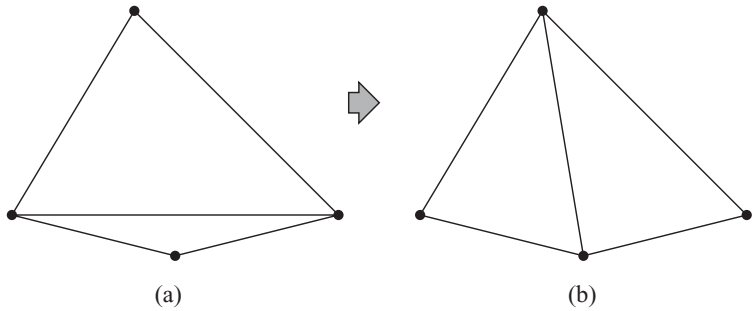


図 4.35: Edge Swap

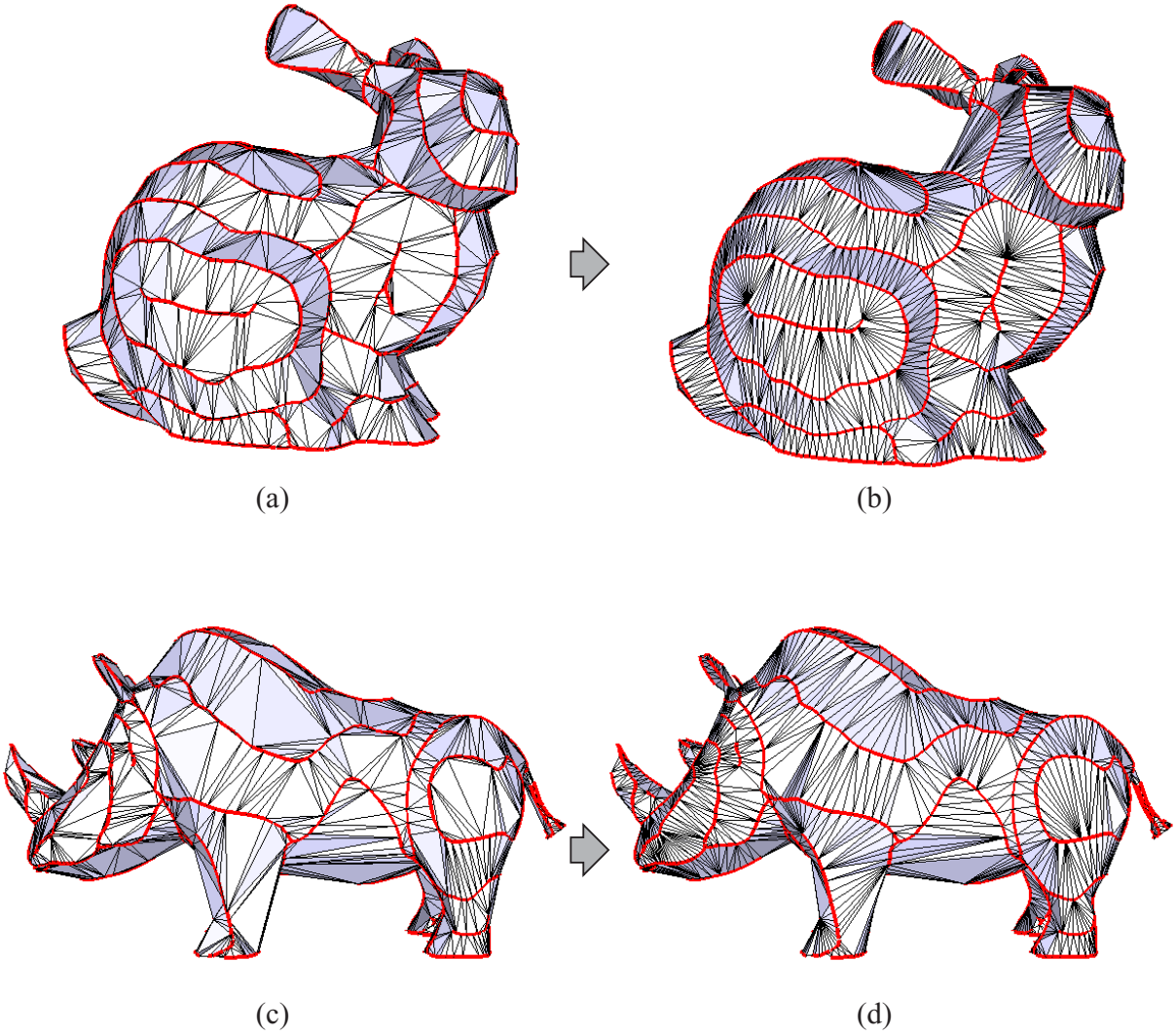
具体的には以下の手法を、各エッジの成す角が改善されなくなるまで、またはループにはまった場合には特定の回数行うまで繰り返し処理した。

1. e に隣接する 2 面の法線の成す角 θ を求める。
2. e に Edge Swap を行い、再度隣接する 2 面の法線の成す角 θ' を求める。
3. $\theta' \leq \theta$ であれば、 2. の Edge Swap を元に戻す。

この結果は図 4.36のようになった。また、エッジに隣接する 2 面の成す角が 150 度よりも小さい辺の数は以下の表 4.4のようになった。このエッジの数が少ないほど、滑らかな STRIP で近似されたことになる。この結果から、 Edge Swap を行うことで、約 2 分の 1 から 3 分の 2 程度の折れ角の小さな辺を減らすことができたことがわかる。ただし、図を見ると、例えばウサギの首の後ろ側など、形が大きく変形している様子が確認でき、形状の近似精度は落ちていることがわかる。

表 4.4: 隣接 2 面の成す角が 150 度よりも小さい辺の数

	処理前	処理後
ウサギ	502	176
サイ	462	246



(a) (b)

(c) (d)

図 4.36: Edge Swap

4.9 手作業で作られたペーパークラフトとの比較

前節までで提案した手法により、人の手を介すことなく計算機による一連の処理によって、メッシュモデルの展開図を自動で生成できるようになった。ここで提案された手法は、なるべく滑らかな形状特徴を維持することを目的に考案されたものである。本節では、この手法によって得られたペーパークラフトが、従来の人の手によって設計されたペーパークラフトとどのような違いがあるかを判断するために、それぞれの作品の比較を行う。

図 4.37(a) は、ウサギのメッシュモデルを元に、ペーパークラフトにすることを目的とした簡略化を手作業によって行ったものである。なお、この簡略化は、なるべく元の形状特徴を維持しながら、ペーパークラフトとして組み立てやすくなるように意識して行ったものである。図中の赤い線は切断する辺として指定した辺を表す。

図 4.37(b) は、切断箇所を強調表示したものであり、図 4.37(c) が展開図である。図 4.37(d) は実際に組み立てを行った作品の写真である[1]。

前節までで提案した手法の結果として得られた作品の写真（図 4.27(d)）と比較すると、はるかに少ないパーツ数で、全体的に荒い近似がなされていながらも、形状特徴はよく保存されていることがわかる。本節で提案する手法によって作成された図 4.27の作品は、各パーツが細い STRIP で近似され、手作業で作ったものよりも元のメッシュに形は近いが、工作はその分大変であることがわかる。

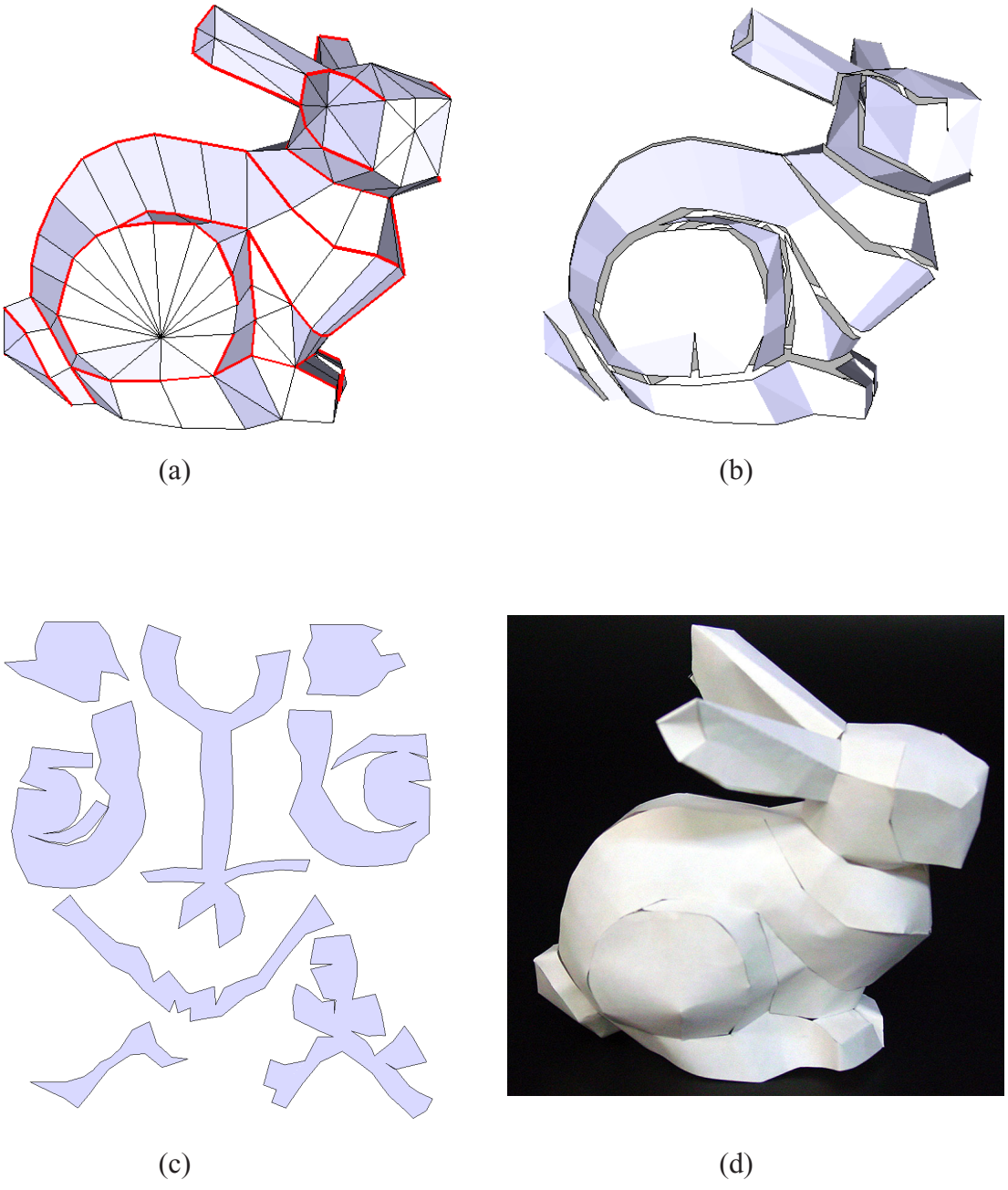
そこで、比較のために、本手法の近似の度合いを表す STRIP 領域の幅 h を、図 4.27で用いたものの 2 倍の値とし、改めて生成したものを図 4.38に示す。図 4.38に比べると、パーツの数が少なくなり、手作業で作成したものと複雑さの程度は比較的近いものとなった。なお、大きさを 15cm としたときに、手作業で作成した展開図に対して算出されたコストは 1988 であり、この展開図のコストは 2074 であった。このことから、工作の手間はどちらも同程度であると判断できる。

手作業によって作成されたペーパークラフトと、ここで生成されたものとを比較すると、次のような異なる点が挙げられる。

- 手作業で作成されたペーパークラフトは、元のメッシュがそれほど鋭角でない場所もメリハリがつけられ、全体的な形の特徴が捉えられている。
- 計算機による生成では、特に頭部の全体的な形状が元のメッシュモデルから大きく変形してしまっている。 STRIP 領域の境界の位置が最終的な形に大きな影響を与えるため、位相的な距離に基づく単純な領域分けでは、適切な位置に境界が配置されない問題が確認できる。
- 手作業と計算機による生成では、耳と頭部、尻尾のパーツ分けは共通しているが、腹部と背部を合わせた体のパーツ分けは異なるものになっている。手作業の場合には、背中に一本通った切れ目が入り、前部から後部にかけてウサギの体の流れに沿ったパーツ分けが行われている。計算機による生成では、最初にパーツ分けを行う際に、細長いパーツは生成されにくいため、全体的に丸い形のパーツの組み合わせで全体が構成される傾向にある。

[1] このときは展開図の表が内側にくるように組み立てたため、実物は画面のものと左右が逆になってしまった。この写真はそれを補正するために左右反転させてものである

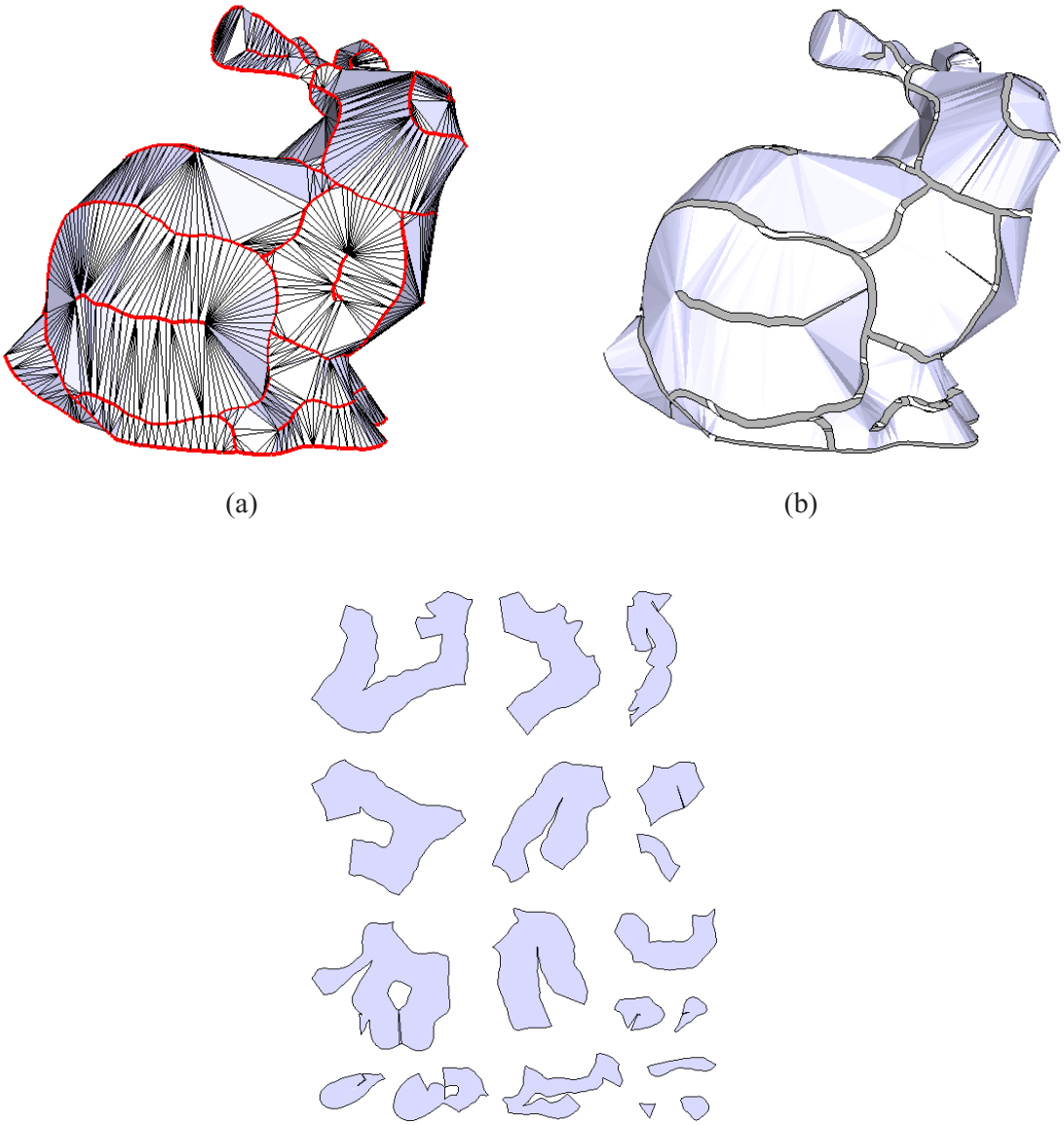
- 手作業で作成した簡略化モデルは、全体を構成する面の数が少なく (236 面)、耳はポリゴンの角ばった特徴などが見て取れるが、実際に工作をしてみると、紙の柔軟性によって丸みを帯びた形状特徴も非常によく維持されている。



(a) (b)

(c) (d)

図 4.37: 手作業で作成したウサギのペーパークラフト



(a) (b)

(c)

図 4.38: STRIP 領域の幅を 2 倍に設定した例

4.10 考察

本章では、そのまま展開したのでは工作できないメッシュモデルを STRIP の集合で近似することで、工作可能な展開図を近似的に生成する手法を提案した。

この手法により、面の数が数万程度の、特に動物のような丸みを帯びたメッシュモデルに対して、滑らかな形状特徴を維持しながら、実際に工作可能な近似的な展開図を生成することができることを確認した。今回の実験で用いたウサギのモデルは測定データから作成されたものであり、サイのモデルは人の手によって編集、作成されたものである。この 2 つは、メッシュの並び方が大きく異なるが、どちらに対しても適切に機能することが確かめられた。

本手法では、最初におおまかにパーツ分けを行って、その後、パーツを STRIP 領域に分割することを行った。この STRIP 領域の境界が切断される辺となり、その内部は簡略化によって内部頂点を持たない滑らかな形となるため、この領域分けの方法が、最終的に得られる形に大きな影響を与える。本手法では、パーツの境界からの位相的な距離に基づいてこの領域分けを行うことで、ドーナツ状の帯の形をした STRIP 領域の生成を行った。そのため、分割に用いる幅によって近似の精度が異なるものとなる。幅が小さい場合は、よりよく形状を近似できるが、その分工作は大変になる。それとは逆に、幅が大きい場合は、近似の精度は低下するが、その分工作は楽になる。

また、 STRIP 領域の境界に乗らない形状特徴や滑らかな丸みを帯びた形状の特徴に対応するために、抽出された特徴線を切断線として追加することや、閉領域から中心線を抽出し、それを切断線として追加する手法を提案した。これにより、元の形状特徴が簡略化後も維持されやすくなった。

さらに、得られた切断線を位相的に、および幾何的に平滑化することで、工作のときに切り出しを行う展開図の輪郭を滑らかにすることを行った。この切断線を固定したままメッシュの簡略化を行うことで、 STRIP の集合で元のメッシュを近似表現することができた。 STRIP の展開図は前章で述べたポリゴンモデルの展開図と同じ手法で生成することができ、また工作の時には折り曲げを省略できるという利点がある。これにより、元のメッシュモデルの滑らかな形状特徴を維持したペーパークラフトを作成できることを、サイとウサギのモデルに対して確かめることができた。

ところで、本章で提案した手法では、予め STRIP 領域の幅を定めて、場所によらずに一律の幅を使用したが、この幅を形状の幾何的な特徴に基づいて、例えば平面に近い領域は広い幅、そうでない領域は細い幅、という具合に動的に変化させられるようにできれば、より効率の良い STRIP 領域への近似が可能と思われる。人が手作業によって作成したペーパークラフトと比べると、 STRIP 領域の幅が位相的な距離にのみ基づいているため、形状特徴がうまく維持できないという問題があるように感じられた。特に、この動的な領域幅の変更は、メッシュの粗密が部位によって異なるモデルを対象とする場合に必要になると思われる。

本手法の問題点の 1 つとして、近似の許容誤差を明示的に指定できないことが挙げられる。 STRIP 領域の幅で近似精度を調整することはできるが、事前に設定された範囲内に誤差が収まるような STRIP 領域の幅を予め決定することはできない。今後の課題として、許容誤差の指定ができるようにすることが挙げられる。また、パーツ分けや中心線の抽出などで必要な各種パラメータの指定は、各処理の結果を見ながら試行錯誤で決定しなければならないため、これらを自動で決定できる仕組みも望まれる。

パーツ分けの方法として、本手法では Lévy の手法を使用したが、この手法には全体的に丸

みを帯びたパーツが生成されやすいという傾向があるため、よりペーパークラフトに適したパーツ分けについても考える必要があると思われる。それには、 Alliez らの手法 [66] のように、メッシュモデルの全体的な「流れ」を意識したリメッシュの手法が有効かもしれない。また、本手法で実装したものは、左右対称な形状に対しても非対称なパーツ分けがなされてしまったため、この対称性の維持も課題の 1 つである。また、パーツ分け後の STRIP 領域への分割について、本手法ではパーツの境界からの位相的な距離を用いて、ドーナツ形状になる STRIP 領域を生成したが、そのほかにも、平行な STRIP の集合で近似することも考えられる。その結果、どのような形状が得られるか、実験してみる価値はあるであろう。

ところで、動物の耳や鳥の翼など、薄い部位を作成する場合、人の手による場合は 1 枚の紙で近似表現してしまうということも行われるため、このような位相の変化を伴う近似表現についても検討する必要があるだろう。

本章で提案した手法は、まだ課題も多く残されているが、メッシュモデルを複数の領域に分割し、それを滑らかな STRIP で近似する、という基本的な考え方は、従来の可展面による近似のアプローチよりも実装が容易であり、様々な形状に対して単純なメッシュ操作だけで完結するという優れた面を持っている。この優れた面をさまざまなペーパークラフトの作成に活かす為に、上記で述べたような、さらなる手法の改善が必要であると思われる。

第 5 章

折り紙建築の設計手法

第 5 章

折り紙建築の設計手法

本章では、折り紙建築の設計を計算機で支援するための手法を述べる。 90 度に開いたときに形が立ち上がるタイプの折り紙建築について、ボクセルを用いることで設計支援と展開図の作成を効率的に行う方法、および平面多角形の集合を用いることで自由度の高い形状の設計を支援する方法を提案する。また、 180 度型の折り紙建築について、既存のポリゴンモデルから断面形状を取得することで、格子状に形が立ち上がるものを設計するための手法を提案する。それぞれの手法について、その理論と応用についてまとめ、実際にアプリケーションを実装して作成した例題を紹介する。

5.1 対象とする折り紙建築

折り紙建築とは2.1.3節で述べたように、3次元形状を所謂ポップアップカードで表現する手法であり、紙で作られた3次元形状を2つ折りで平面へ折り畳むことができるという特徴をもつ。また、折り畳まれた台紙を再度開くことで、そこに3次元形状を再現できる。この独特な特徴のために、意図した形状が立ち上がる折り紙建築を設計するのは容易ではなく、従来は熟練者の手による試行錯誤でその展開図の作成が行われてきた。また、折り紙建築は作成して台紙の開閉をしてみるまで、実際に意図した形が立ち上がり、きちんと折り畳むことができるかどうかを確認することができないため、設計の試行錯誤は非常にコストがかかる作業である。

そこで本章では、この折り紙建築の設計を計算機で支援する手法を提案する。

折り紙建築には、製作者の創意工夫により90度、180度、360度に開くと形が立ち上がるものや、1枚の紙から作成されるもの、複数のパーツを組み合わせて形を作るものなど、様々なタイプのものが存在する[67][68]。全てのタイプを計算機上で統一的に扱うのは困難であるため、ここでは特に折り紙建築の基本的な特徴を持った、1枚の紙から作成され90度に開いたときに形が立ち上がるタイプのもの(**90度型**)と、格子状に組み合わせた立体が180度に開いたときに立ち上がるもの(**180度型**)を扱うこことする。

また、一般的なポップアップカードは複数の紙のパーツを貼りあわせて形を作る場合が多いが、ここで扱う90度型の折り紙建築は、1枚の紙から切り込みと折り曲げを追加するだけで作成するものとする。このように1枚の紙だけから作成する折り紙建築は文献[21]に多く紹介されている。パーツの貼り合せを行わないため、作成できる形の制約が大きい反面、切り抜きと折り曲げだけで作成できるので工作が容易であるという利点がある。一方で、180度型のものは複数のパーツを組み合わせないと立体を表現できないため、本章で紹介する180度型のものは、立体の断面形状を格子状に組み合わせることで作成するものを対象としている。

本章の5.2節では、この90度型の折り紙建築を**ボクセル表現**を用いて扱う手法を提案する。続く5.3節では、このボクセル表現を用いた手法を応用して、既存のポリゴンモデルから折り紙建築を自動で作成する手法を提案する。

5.4節では、平面多角形の集合で折り紙建築を表現することで、ボクセル表現では実現できなかった自由度の高い折り紙建築を設計する手法を提案する。

最後に5.5節では、複数のパーツを組み合わせて作る180度型の折り紙建築を既存のポリゴンモデルから作成する手法を提案する。

5.2 ボクセルを用いた 90 度型折り紙建築の設計手法

本節では、ボクセル表現によって 90 度型の折り紙建築の形状データを保持し、 CG によって立体表示を行う手法、及び展開図を生成する手法を提案する。また、この形状データを効率よく作成するためのインターフェースを PC 上に実装し、実際にこのシステムを用いて折り紙建築の作成を行った。

5.2.1 折り畳める条件

本節では 90 度に開いたときに 3 次元形状が立ち上がり、二つ折りに畳める折り紙建築を対象とする。図 5.1(a) はその最も単純な形の例で、 (e) はその断面である。ここで、 (b) 〜 (d) は (e) に似ていながら、実際には折り紙建築には不適切であることを示す。図中の辺の長さを表すスカラ値 a 〜 d について、 1 枚の紙から作成できる条件は

$$a + b = c + d \tag{5.1}$$

で表せる。また、二つ折りに畳める条件は

$$a + d = b + c \tag{5.2}$$

で表せる。 (b) は式 5.1を満たさないために 1 枚の紙から作成できず、 (c) は式 5.2を満たさないために二つ折りに畳められない。 (d) はどちらの条件も満たさない。 (e) は両方の条件を満たすため、本研究で対象とする形状として妥当であり、断面は式 5.1、 5.2より、 $a = c, b = d$ を満たす長方形となることがわかる。

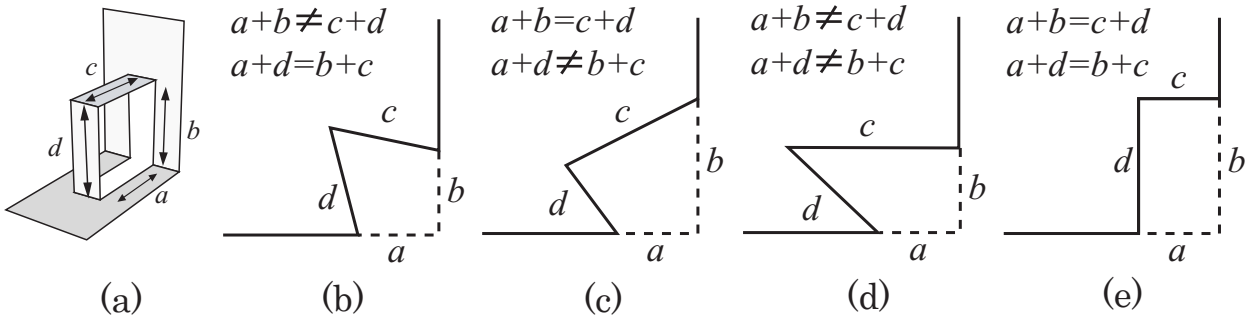


図 5.1: 折り紙建築モデルの側面図

図 5.2は断面が長方形になる形状の例で、 (a) は直方体、 (b) は四角錐の形が 90 ° に開いたときに立ち上がる（長方形の側面および四角錘の底面には紙の面が存在しない）。それぞれ [21] において、「平折り」「斜折り（はすおり）」という名称で紹介されている。これらの基本的な形状を組み合わせることで、より複雑な形を構築することができる。

本節では特に図 5.2(a) で示される「平折り」の集合によって表現される形を扱う対象とする。

5.2.2 ボクセルモデルによる形状表現

ボクセル表現とは 3 次元格子状に配列された単位立方体（ボクセル）の集合で立体形状を表現する手法であり、 CG の世界ではソリッドモデルを表現する際に、境界表現と共に広く用い

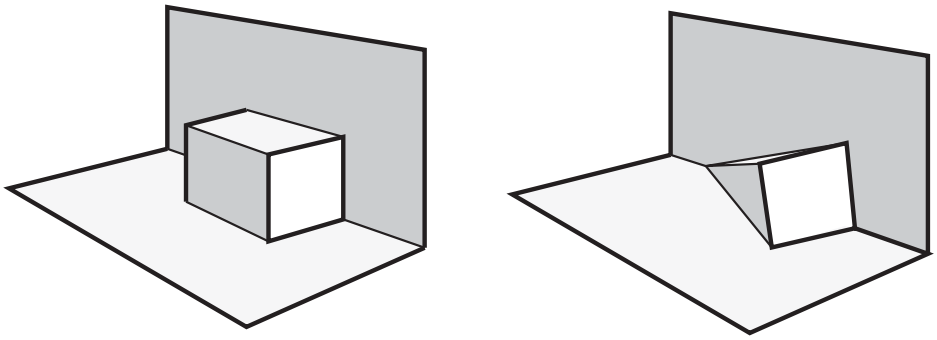


図 5.2: 1 枚の用紙から成り立ち二つ折りに畳める折り紙建築の例

られている [69]。折り紙建築の形状が「平折り」の集合から構成される場合、表現される立体形状は直方体の集合であるため、この折り紙建築が定義する形状を量子化することで、ボクセルモデルによって近似的に表現できる（図 5.3(b)、(d)）。また、このボクセルモデルの正面と上面を抽出し、底面と背面となる台紙を併せれば、ボクセルモデルを折り紙建築として CG 表示できる（図 5.3(a)、(c)）。従って、ボクセルモデルを用いることにより、対象とする形状データを計算機に保持すること、及び折り紙建築モデルを CG 表示することの両方を簡単な実装で行える。

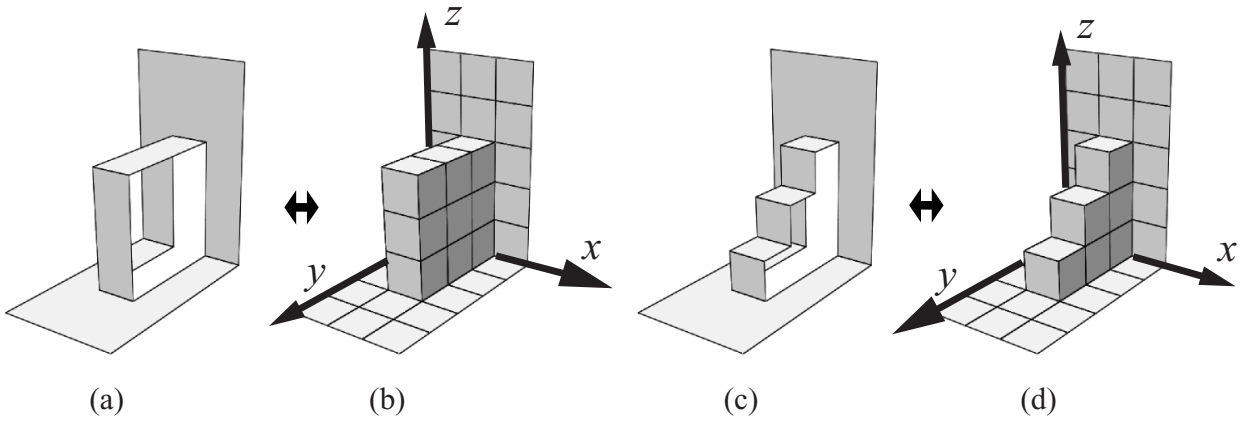


図 5.3: ボクセル表現と折り紙建築モデル

単純な形状の表現

図 5.3のように座標軸をとると、ボクセルモデルのデータは立体の内外を表す 1 ビットの値（内部:1, 外部:0）を、 x, y, z の 3 次元に格納した配列

$$\mathrm{cell}[x][y][z] = \{0, 1\} \tag{5.3}$$

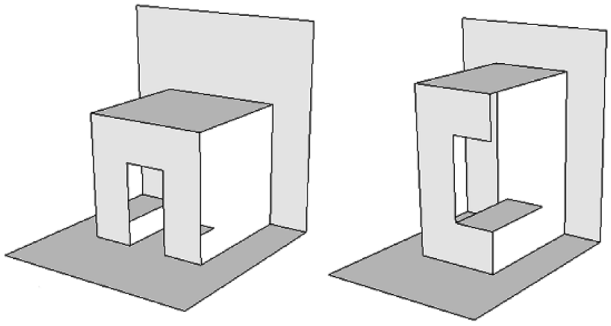
で表現できる。ただし図 5.2(a) の平折りの集合で構成されるため、ボクセルモデルは図 5.3のような階段形状である必要がある。この条件は次のように表される。

$$\begin{aligned} &\mathrm{cell}[x][y][z-1] = 1 \ \textit{and} \ \mathrm{cell}[x][y-1][z] = 1 \\ &\qquad \textit{if} \ \mathrm{cell}[x][y][z] = 1 \ (x \geq 0, y > 0, z > 0) \\ \\ &\mathrm{cell}[x][y][z+1] = 0 \ \textit{and} \ \mathrm{cell}[x][y+1][z] = 0 \\ &\qquad \textit{if} \ \mathrm{cell}[x][y][z] = 0 \ (x \geq 0, y \geq 0, z \geq 0) \end{aligned} \tag{5.4}$$

なお、これ以降では式 5.4を**折り紙建築制約式**と呼ぶ。

開口部が存在する形状の表現

折り紙建築制約式を満たすボクセルモデルは図 5.3(a) の要素から構成される「階段形状」に限定されるため、表現できる折り紙建築の形は大きく制限される。そこで、文献 [21] で「窓」として紹介されている図 5.4(a)、 (b) のような開口部をもつ折り紙建築も扱えるように拡張する。



(a) (b)

図 5.4: 開口部の存在する折り紙建築

図 5.4に示すような、正面からの開口部を持つ折り紙建築のデータを保持するには、式 5.3の形状データに加え、背面（xz 平面）上に開口部の有無を表す情報があればよい。これは 1 ビットの値（有り:1, 無し:0）を x, z の 2 次元に格納した配列

$$\mathrm{isHole}[x][y] = \{0, 1\} \tag{5.5}$$

で表現できる。以降この配列で表現される開口部の情報を**開口部情報**と呼ぶこととする。なお、上面からの開口部も同様にして扱うことができるが、ここでは正面からの開口部のみを対象とすることとする。

この開口部情報を元に、図 5.4のように開口部が存在する折り紙建築を CG 表示するには次のようにすればよい。

あるボクセル (cell[i][j][k]) の正面が境界である時 (cell[i][j][k] = 1 *and* cell[i][$j+1$][k] = 0) に、そこが開口部である（isHole[i][k] = 1）場合には、 cell[i][$j-(k-h)+1$][h] で表現される

ボクセルの上面を表示する（図 5.5(a)）。ここでの h は isHole[i][z] = 0 を満たす、k より小さい最大の z である。図 5.5(a)、(b) ともに $h = 0$ である。なお、$j - (k - h) + 1$ がゼロより小さくなった場合は、さらに折り返しを行い、背面の [i][$k - j$] を表示する（図 5.5(b)）。

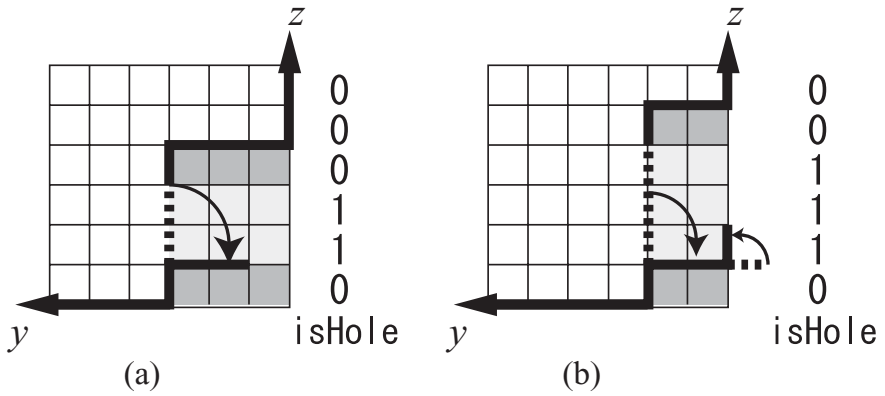


図 5.5: 開口部を持つ折り紙建築の CG 表示方法

5.2.3 計算機による設計支援

展開図の生成

ボクセルモデルで表された図 5.6(a) に示すような折り紙建築を作成するには図 5.6(c) のような展開図が必要である。この展開図を、折り紙建築の形状を表現するボクセルデータから生成することを考える。

対象とする折り紙建築は 1 枚の紙から作成されるため、折り紙建築の面と展開図を構成する領域は図 5.6(a) と (b) の a〜l に示すように、ボクセルの各面と展開図中の正方領域（**セル**）が 1 対 1 に対応する。(a) の a〜l で表される各ボクセルの境界面について、折り紙建築モデル上での隣接セルの有無、及び成す角を調べることで、展開図 (b) 上に現れる山折り線、谷折り線、及び切断線を決定することができる。成す角と線種の対応は表 5.1の通りである。表をもとに、対応する線分を配置することで展開図を得ることができる。この展開図生成の手法は開口部が存在する場合にも有効である。

表 5.1: 隣接ボクセル間のなす角と線種

なす角	90 °	180 °	270 °	なし
線種	谷折り	なし	山折り	切断

折り紙建築として実現できる形の判定

開口部を作成することで表現できる形の幅が広がるが、それと共に実際には折り紙建築として工作できない形も作成され得るという問題がある。図 5.7は折り紙建築として適切でない場合の例である。

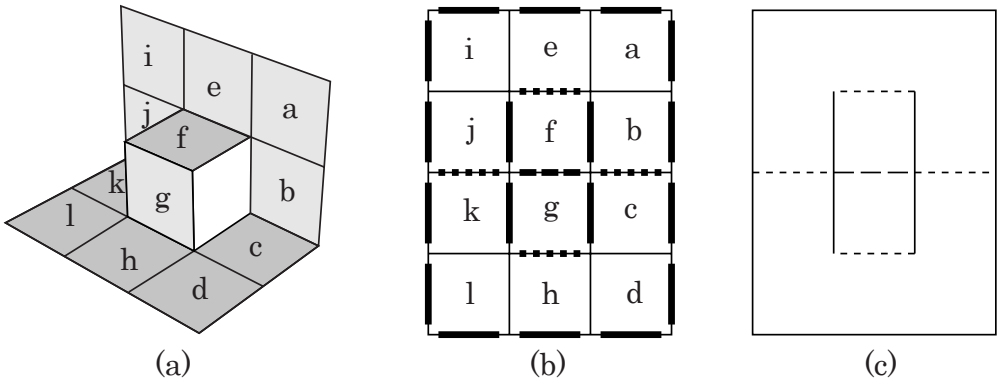


図 5.6: 折り紙建築モデルと展開図

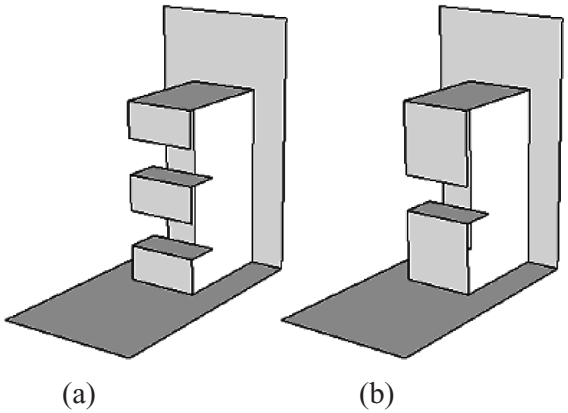


図 5.7: 実現不可能な開口部をもつ場合

図 5.7(a) には宙に浮いた部分が存在する。 (b) は下方と上方から立ち上がっている部分が分離しているため、閉じた状態から開いた時に引き起こされない。これらの形状は折り紙建築として妥当ではないため、計算機内で保持しているデータの妥当性をチェックする仕組みが必要となる。

ここでは、宙に浮いている箇所がある場合、および上下からの支えが無い 2 回以上の折り曲げが存在する場合（開口部の作成に必要な 1 回の折り曲げは許容する）に、形状が妥当でないものとする。なお、上下からの支えがあるというのは、折り紙建築を構成する各面の上辺、および下辺を介して接する面を再帰的に辿っていたときに、どちらも台紙面に到達する場合をいうこととする。このような妥当性の判定は、展開図を作成後、次のようなアルゴリズムによって形状のチェックを行うことができる。

1. 展開図の各セルについて、以下の方法で巡回を行い、巡回済みのフラグ立てを行う。

   (a) 展開図の左上隅から開始し、隣接する左、下、及び右のセルを再帰的に巡回する。ただし隣接するセルの間に切断線が存在したら巡回しない。

   (b) 展開図の左下隅から開始し、隣接する左、上、右のセルを再帰的に巡回する。ただし隣接するセルの間に切断線が存在したら巡回しない。

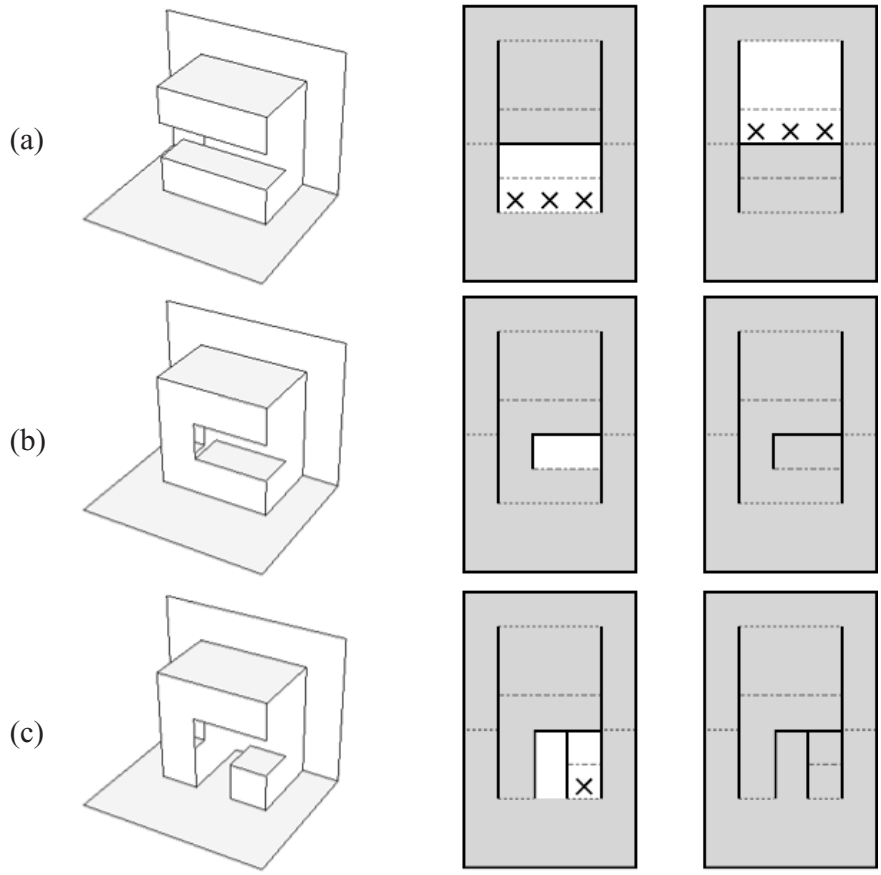


図 5.8: 妥当性の判定

2. 展開図の中の正面を成すセルについて、 1 の (a) と (b) のフラグのどちらか一方、または両方が立っていないセルが存在したら折り紙建築として適切でない形状である。

上記のアルゴリズムを適用した例を図 5.8に示す。展開図のうち、左側が左上隅から巡回を開始したもの、右側が左下隅から巡回を開始したものである。 (a)、 (c) はフラグの立っていない正面のセル（図中の×印）が存在するため、適切でない形状であると判定される。

エディタの実装

折り紙建築制約式を満たすボクセルと開口部情報があれば、それを折り紙建築として作成可能なことを前節までで示した。これらの情報を簡単に作成できれば、一般のユーザでも折り紙建築の設計を行えるようになる。そこで、対話的かつ簡便なインターフェースで折り紙建築モデルを構築するためのエディタを実装した。

エディタでは、カーソルを移動させて、目的位置でボクセルの追加と削除を行える。この操作について、次のような実装を行うことで、生成されるボクセルが常に折り紙建築制約式を満たすことを保証できる。

- ボクセルの追加操作（図 5.9(a) → (b)）

cell[i][j][k] にボクセルが追加された時は、自動的に cell[i][y][z] = 1 ($0 \leq y < j$ *and* $0 \leq z < k$) とする。

- ボクセルの削除操作（図 5.9(c) → (d)）

  cell[i][j][k] のボクセルが削除された時は、自動的に cell[i][y][z] = 0 ($y \geq j$ *and* $z \geq k$) とする。

エディタはボクセルの編集と開口部の指定が行えるようになっており、リアルタイムに CG 表示を行うことで、ユーザは折り紙建築の完成様子を確認しながらモデルを作成してゆくことができる（図 5.9）。

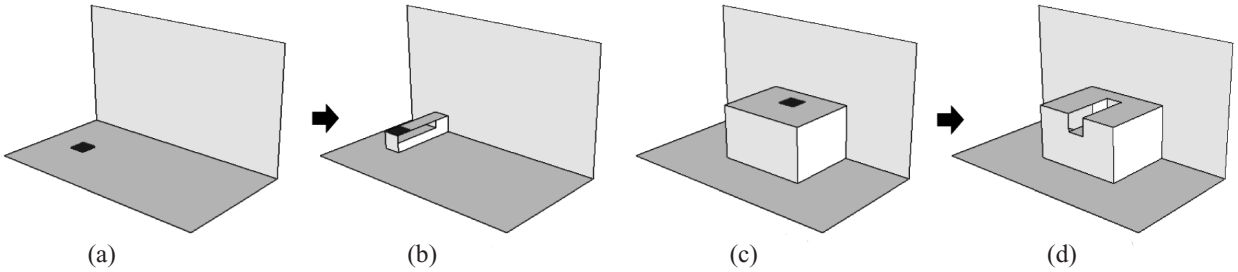


図 5.9: エディタによる編集
(a) → (b): ボクセルの追加 (c) → (d): ボクセルの削除
（黒い四角形はカーソル）

CG アニメーションによる開閉シミュレーション

ボクセルモデルは一般に単位立方体の集合で表されるが、この単位立方体の各頂点 (x, y, z) を、次式で変換することで折り紙建築の開閉途中の形状を表現できる。

$$\begin{cases} X = x \\ Y = y - z\cos\theta \qquad (0^\circ \leq \theta \leq 180^\circ) \\ Z = z\sin\theta \end{cases} \tag{5.6}$$

式 5.6の θ の値を徐々に変化させることで、折り紙建築の開閉をアニメーション表示できる。図 5.10は実際に CG でアニメーションを作成したものの一部である。 1 枚の紙から作成され、90 度の時に形が立ち上がり、二つ折りで平面に折り畳めることが確認できる。

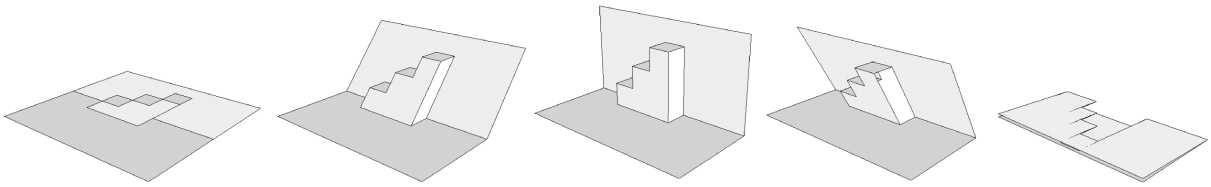
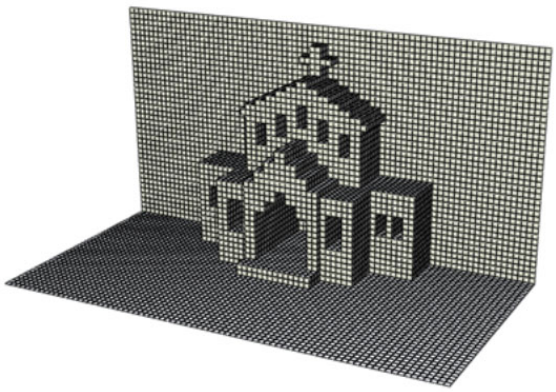


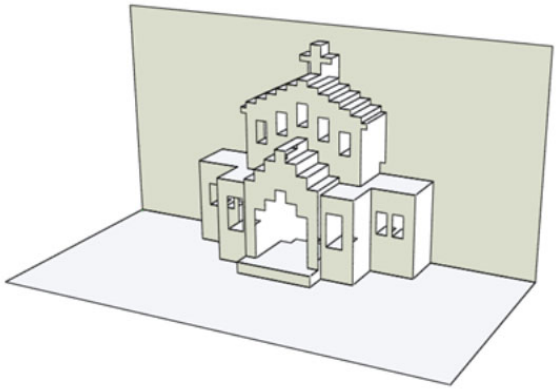
図 5.10: 折り畳み途中の形状表示

5.2.4 結果

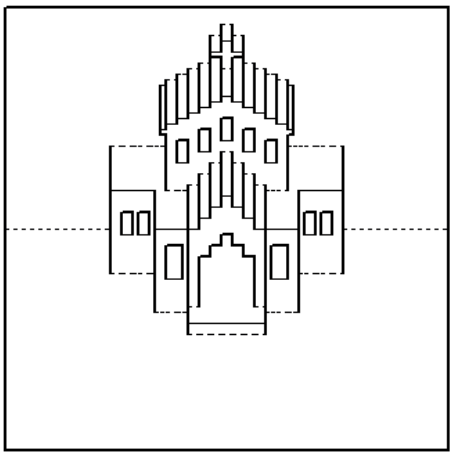
本節で提案した手法を実装し、折り紙建築モデルを対話的インターフェースで作成した結果を図 5.11に示す。 (a) は作成したボクセル表現の建築物モデルであり、開口部情報の存在する箇所はボクセルを非表示としている。 (b) は (a) のボクセルモデルで表される折り紙建築を CG 表示したものである。 (c) は 5.2.3節で述べた手法で生成した展開図であり、 (d) はその展開図を元に工作用紙で作成した折り紙建築である。ボクセルの解像度は奥行きと高さが 40、横幅が 80 であり、インタラクティブにモデルの構築と CG 表示を行えた。モデルの作成に要した時間は 30 分程度である。



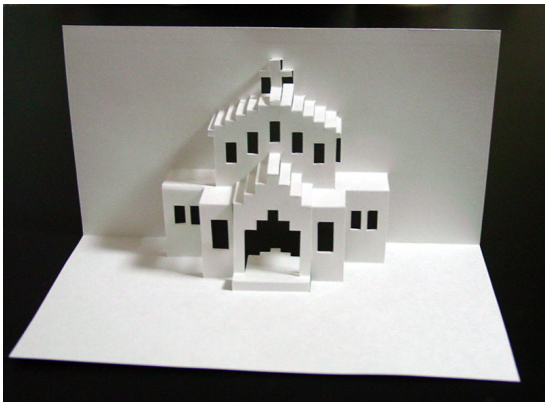
(a)



(b)



(c)



(d)

図 5.11: 対話的な折り紙建築の作成

5.2.5 教育利用

宮城県仙台市立田子中学校のご協力を得て、1 年生の技術・家庭の授業の一部としてこの手法を実装したアプリケーションを活用していただいた。ここでは中学校の技術・家庭における「ものづくり」の学習の一部をターゲットとし、紙工作の授業を行った。なお、授業においては「折り紙建築」ではなく「ポップアップカード」という言葉を用いた。また、システムの機能拡張として、以下の 2 点を行った。

- 下記製作図の学習内容に合わせて、第三角法による正投影図が表示できるようにした。3 次元の CG だけでなく、図面を出力できるということが重要である。
- モデルを WEB 上で公開するために、作品を立体表示し、またその展開の様子をアニメーション表示できる専用のビューワーを開発した。これは、コンピュータ上でデザインした作品を、アニメーションというより面白い形で表し、それをお互いに公開しあい、また外部に対して情報発信することによって生徒のモチベーションを高めようとするためのものである。

授業は延べ 6 回行われた。表 5.2に各回の授業内容を示す。

表 5.2: 授業の内容

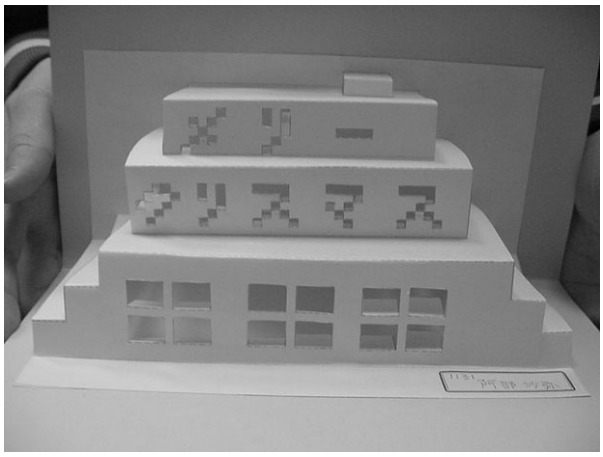
1 回	ポップアップカードの説明。（パーソナルコンピュータは使わずに）簡単なポップアップカードを実際に作らせてみて、その仕組みを理解させる。
2 回	パーソナルコンピュータ上でアプリケーションの使い方や印刷の仕方を説明。
3 回〜 6 回	パーソナルコンピュータでポップアップカードをデザインし、実技としてそれをカッターで切って組み立てさせた。課題としては、アルファベットの文字と立体（建築物など）を与えた。最後に作品の展示を行い、また、ホームページにおいてパーソナルコンピュータで作ったモデルの表示と工作したカードの画像を公開し、さらに投票によってコンテストを行った。

図 5.12に授業の様子を示す。図 5.12(a) のように、生徒は一人一人パーソナルコンピュータ上のアプリケーションを用いてデザインを行った。PC を使う前に、紙にデザインをするように指導してあり、そのデザインを見ながら入力する。図 5.12(b) は生徒の作品例である。生徒の作品は校内に展示され、父兄にも公開された。また、WEB 上でも公開され、人気投票コンテストなども行われた。このように授業は盛り上がりを見せた。

このアプリケーションを用いた授業について、次のような評価を得た。

- 最初はもっとも基本的な機能だけを紹介し、その後、グリッドの大きさを変えたり、背景の色を変えるなどの操作を段階的に紹介することは必須である。また、機能を小出しにすることは、向上心を刺激することにもなる。
- システムのユーザーインタフェースは簡明で習熟は全く問題がなかった。
- パーソナルコンピュータを使うこと自体は日常化しており、大きな動機付けとはなっていないが、クラスメートと同じ課題に取り組むことによって、作品はもちろんのこと、画面の色の工夫など、お互いに競い合うようにして熱中していたようだ。

(a)



(b)

図 5.12: 教育現場での活用

5.2.6 考察

本章では、平折りと窓の集合で表現される 90 度型の折り紙建築について、ボクセルを用いることでデータの保持、 CG 表示、及び展開図の生成を容易に行えることを示した。ここで提案した手法を用いることで、特に折り紙建築の知識を持たない中学生など、ごく一般のユーザーが折り紙建築の設計を容易に行えるようになった。

しかしながら、本手法で作成できる折り紙建築の形状はボクセルで表現できる形状であり、また平折りと窓の集合で表現できるものに限られているため、作成できる形の自由度はそれほど高くない（この問題を解決するための別の手法は 5.4節で述べる）。

また、教育利用については、次のような知見が得られた。

教育効果

本手法を実装したアプリケーションでは、まず生徒はソフトウェアによって 3 次元立体を仮想世界でデザインする。ここでは、 3 次元 CG によるゲーム的な画像の面白さを体験するとともに、ものを作る過程での試行錯誤を仮想世界において自由に行うことができる。通常の工作では、限られた時間やコストのために、試行錯誤が非常に限られており、「失敗することができない」という、工作への動機付けを失わせる一つの要因を回避できたといえる。

そして、デザインが終了すると、その立体を実体化するための材料が紙という素材として出力され、生徒はそれを工作する。既成のデザインではなく、自分がデザインしたものが実体化されることの達成感を与えることができ、よりもの作りへの動機付けを行えると考えられる。

また、コンピューター上ではいくらでも複雑なものができるが、実際に実技ではギブアップする例があり、構想段階から現実の工作を意識するということが訓練される。とかく TV ゲームなどと違い現実の制約を考える重要性が指導できる。

3 次元認識能力の開発

3 次元空間における、形や物と物との位置関係、奥行きなどを把握する空間認識能力は基本的な能力の一つである。技術・家庭においては、構想図や製作図の書き方を指導する過程で、3 次元物体を認識し、 2 次元の図で表現するという課題を扱っているが、正確に表現を行える能力は教科書の図などの平面的な教材では涵養しにくいと考えられる。本教材での 3 次元 CG によって、様々な方向から物体を回転して見ることができ、また、ポップアップカードが展開して行く過程をアニメーション表示することによって 3 次元形状の認識が容易にできる。さらに、その 3 面図も同時に表示できるので、 3 次元と 2 次元の関係を把握する能力が効果的に涵養できると期待される。そして、紙工作によって実体を手に取ることによってさらにそれを体験することができる。

新しい図工教材分野の開拓

教育現場での試行を通じて、提案するシステムを教育現場に適したものへと改良し、マニュアルや素材なども充実して、誰でもが簡単に利用できる教材として提供することができた。これによって CG と工作との融合によりもの作りの楽しさを味わうという新しい教育法や教材分野の可能性を提示し、同種の教材ソフトウェアの開発や、将来の教育現場への普及が期待できると言える。

5.3 ポリゴンモデルからの 90 度型折り紙建築の自動生成

前節では折り紙建築制約式を満たすボクセルデータと開口部情報が存在すれば、折り紙建築の CG 表示と展開図の作成を行えることを示した。ところで、 CG の世界ではポリゴンモデルを用いた 3 次元形状の表現が一般的であるため、これらの既存のデータを活用できれば作業コストを大きく軽減できる。そこで本節では既存のポリゴンモデルから折り紙建築を作成する方法を述べる。

5.3.1 ポリゴンデータのボクセルへの変換

ポリゴンデータからボクセルを生成する手法には、 Chen ら [70] による高速変換のためのアルゴリズムなどが研究されているが、折り紙建築で扱う形状はその性質から側面の情報を持たず、底面と背面は平面であるという特徴がある。そのため、正面から投影した各セルの奥行き値が取得できれば十分である。 OpenGL ライブラリ [71] を用いて対象とするポリゴンモデルを正面から正射影でレンダリングすると、そのレンダリング結果から各ピクセルごとの奥行きを表す Depth 値を取得できる。そこで本手法では OpenGL を用いて Depth 値の取得を行い、ポリゴンモデルをボクセルモデルへ変換した。

5.3.2 折り紙建築制約を満たすボクセルと開口部情報の生成

前節の手法で得られたボクセルデータは必ずしも折り紙建築制約式を満たすとは限らない。そのため、元の形状特徴をなるべく維持しながら制約式を満たす形に変形させ、それと同時に開口部情報の生成を行う。具体的には次の (a) 〜 (c) の処理を行った。

(a) 微小凸凹の除去

前節で述べた手法で生成したボクセルデータを折り紙建築制約を満たす形状に変換する際、小さな凹凸がボクセルモデルの全体の形状に影響を与えることがあるため、前処理として微小凹凸の除去を行う。そのための手法として、 yz 平面による各 x 座標値の断面について、 Erosion と Dilation という 2 つのモルフォロジ処理を組み合わせた Opening 及び Closing 演算 [72] を行う。この演算は 2 値画像の特徴抽出などで用いられる集合操作の 1 つで、微小な凹凸を除去する性質を持つ。本手法では演算の構造要素として、対象ボクセルを中心とする 3×3 四方のボクセルを使用した（このサイズを変更することで除去される微小凹凸の程度を調整できる）。この処理によって微小凹凸が除去される様子を図 5.13に示す。

(b) ボクセル追加による制約充足

(a) の処理の後、折り紙建築制約式を満たすように、ボクセルを追加する処理を行う（図 5.14）。この処理は次式を z の値について降順に行うことで達成される。

$$\mathrm{cell}[x][y][z] = 1 \; if \; \mathrm{cell}[x][y][z+1] = 1 \tag{5.7}$$

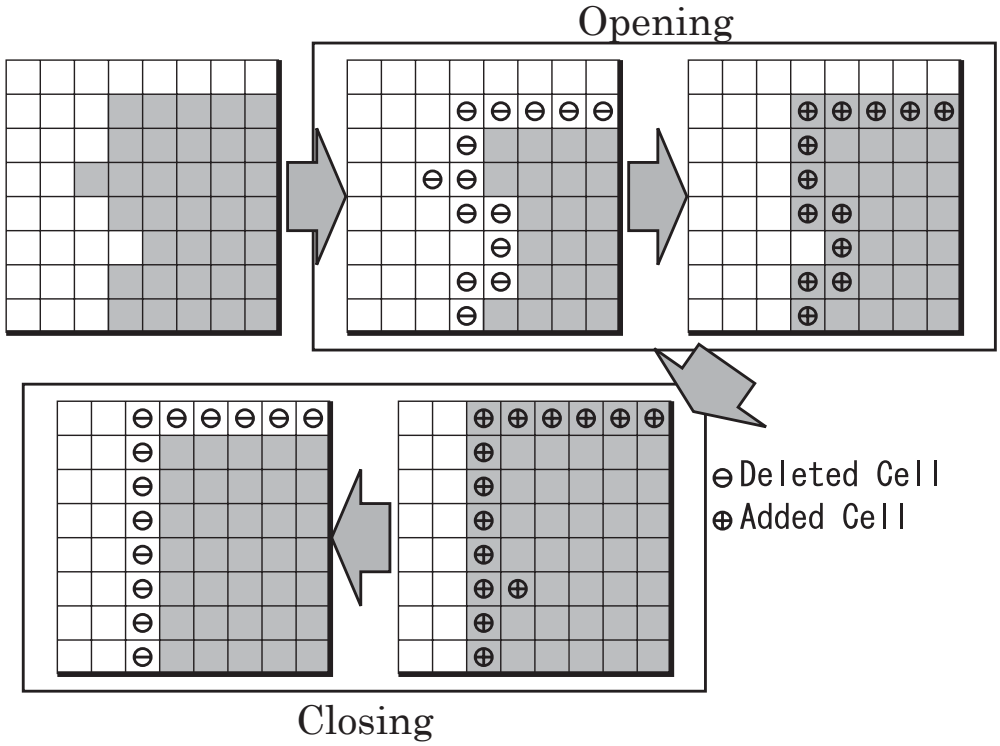


図 5.13: Opening と Closing 演算による凹凸の除去

(c) 開口部情報の生成

(b) の処理を行う際に、ボクセルの追加を行った箇所に isHole フラグ（式 5.5）を設定する。フラグを 1 つ立てる毎に 5.2.3節で述べた形状の妥当性チェックを行い、折り紙建築として不適切な形状になる場合には、フラグ設定を解除する。

5.3.3 幅の細い上面の省略

座標平面に平行でない面がポリゴンモデルに含まれる場合、本手法では座標平面に平行な面で近似せざるを得ないため、非常に多くの面が生じてしまう問題がある。幅の細い面が多数存

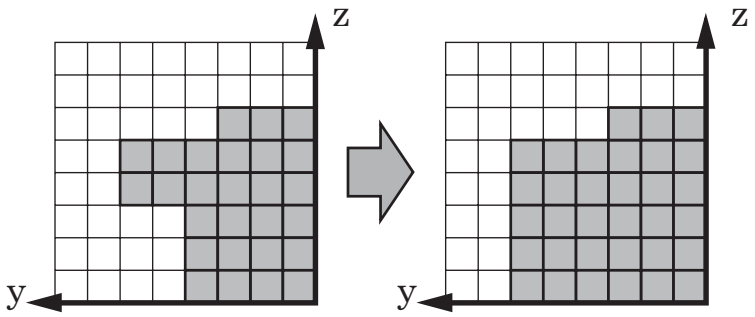


図 5.14: 折り紙建築制約式を満たす形への変換

在する場合、 CG 表示ができても実際に工作することが現実的でない場合がある。そこで代替手法として、図 5.15(a) を (b) に示すような、幅の細い上面を省略した形状で出力し、工作にかかる作業を軽減する手法を提案する。

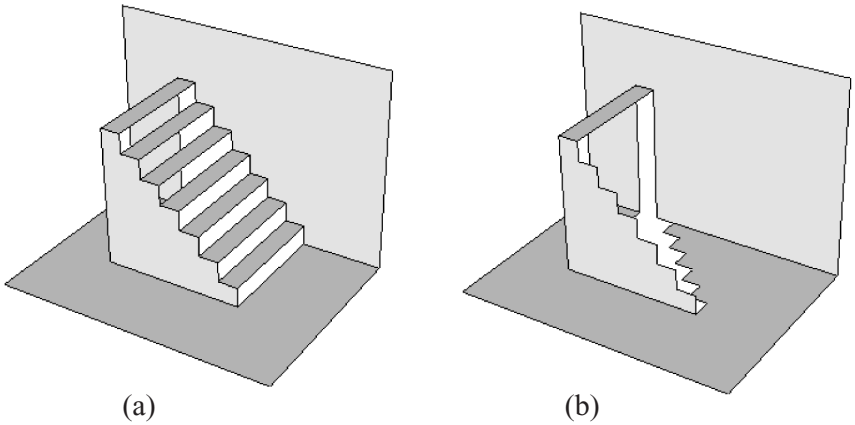


図 5.15: 幅の細い上面を省略した例

省略する上面は、幅が閾値（図 5.15では 1 ボクセル分）以下であり、省略しても折り紙建築として妥当な形を保つものである。例えば図 5.15(a) の全ての上面を省いてしまうと 90 度に開いたときに正面部が立ち上がらない。この妥当性の判定は 5.2.3節で用いた手法で行える。図 5.16(a) は閾値を 1 ボクセル分に設定した場合の正面図の例である。太線は上面を生成する箇所を表す。 b, c, d の列はそれぞれ幅が 1 の上面を持つが、支えを残すために最も高さのある b は省かない。 f, g, h の列は上面の幅が閾値より大きいため何も行わない。上面を省略した箇所を CG 表示する際には、本来上面に使用されるべきであった面を図 5.16(b) に示すように背面へ移動する。移動先が底面より下に移動するようであれば 5.16(c) のように、再度折り返すことで底面を表示するようにする。これは、 5.2.2節で述べた開口部の CG 表示と同等の方法である。

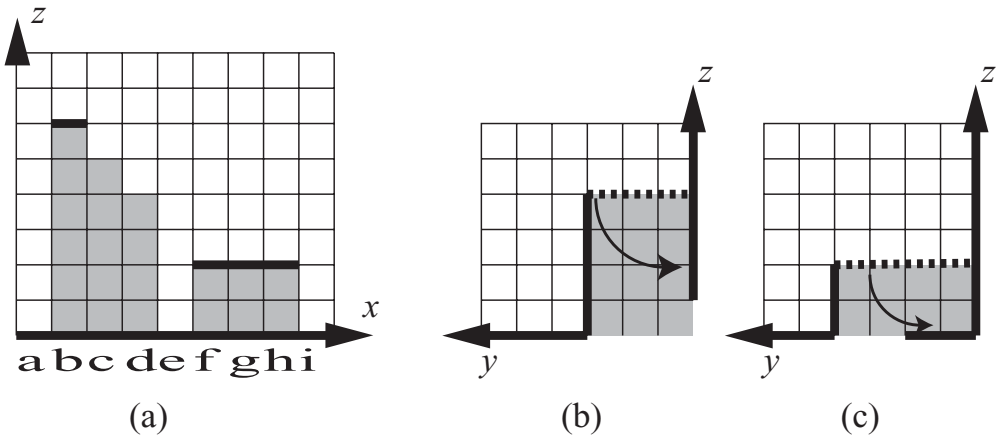


図 5.16: 上面の省略
（(a) 正面図 (b), (c) 側面図）

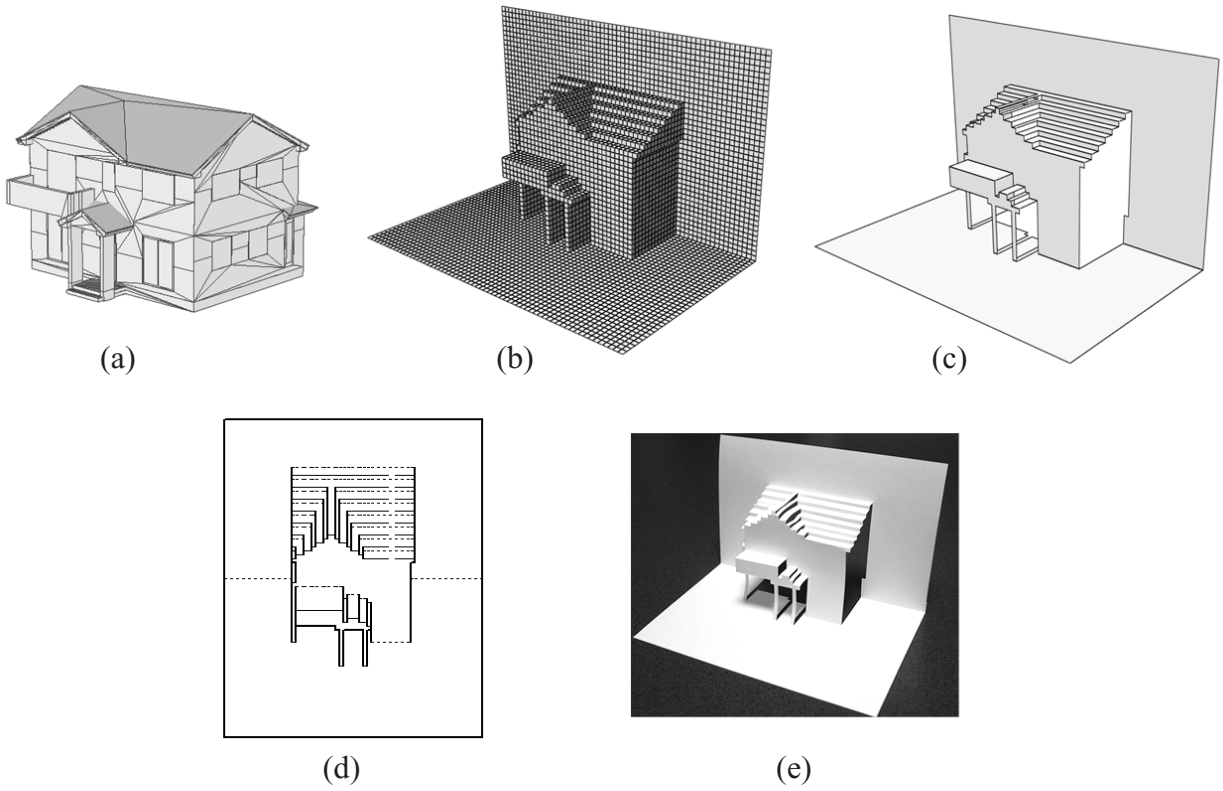


図 5.17: ポリゴンモデルからの作成

5.3.4 結果

既存の建築物のポリゴンモデルから本システムを用いて折り紙建築模型を作成した結果を図 5.17に示す。 (a) は入力に用いた建築物のポリゴンモデル、 (b) は (a) の前方半分をボクセルモデルに変換し、 5.3.2節で述べた手法で折り紙建築制約式を満たすように処理したものである。 (c) は折り紙建築を CG 表示したものであり、 (d) は生成された展開図である。 (e) は (d) を工作用紙で作成した作品の写真である。ボクセルの解像度は奥行きと高さが 40、横幅が 80 であり、微小凹凸を除去する処理には 60 ミリ秒、折り紙建築制約を満たすボクセルへの変換と開口部情報の生成には 2.5 秒かかった。

図 5.18は 5.3.3節で述べた幅の細い上面の省略を行った様子である。ボクセルの解像度を高めて斜線部と曲線部の近似精度を高めた場合でも工作にかかる手間を大幅に軽減できる様子を確認できる。

5.3.5 考察

本章では、既存のポリゴンモデルをボクセルモデルに変換することで、折り紙建築を自動で作成する手法を提案した。また、幅の細い上面を省略することで、工作の手間を軽減する手法を提案した。

折り紙建築の性質上、背面や底面、及び左右の側面の形状は表現できないため、既存のポリゴンモデルに存在するこれらの形状特徴は失われてしまう。また、ボクセルに変換することで、座標軸に平行でない辺や面の特徴は失われやすく、さらに折り紙建築として表現可能な形

への変換により、元の形状からはだいぶ異なったものになってしまう傾向がある。自動的にボクセルが生成される機能は、ユーザーの手間を省く上では有用であるが、それをそのまま折り紙建築の作品にしても、元のポリゴンモデルの形を維持するのは難しいように思われる。本手法は、大まかな形状を簡単に生成するために使用し、それにユーザーが手直しをすることで、最終的な作品を完成させる方法が適しているように思われる。ボクセル表現を用いるという制約と共に、折り紙建築自体の制約もあるため、元の形状の特徴をデフォルメして表現するなどの工夫が必要であるように感じられた。

また、幅の細い上面を省略することにより、工作の手間は軽減できるが、これらの幅の細い上面の集合によって表現されていた「水平でない面」が消えてしまうため、このような面の存在が形状表現の上で重要である場合には望ましくない結果になってしまうと考えられる。「折り紙建築」という名の示すとおり、扱う形状は「建築物」のように、形状を構成する面が座標平面に平行なものが多く、特に正面の形状が全体的な形状特徴を示すようなものが適していると思われる。このような形に対しては、本章で提案した手法も有効に機能するものと思われる。

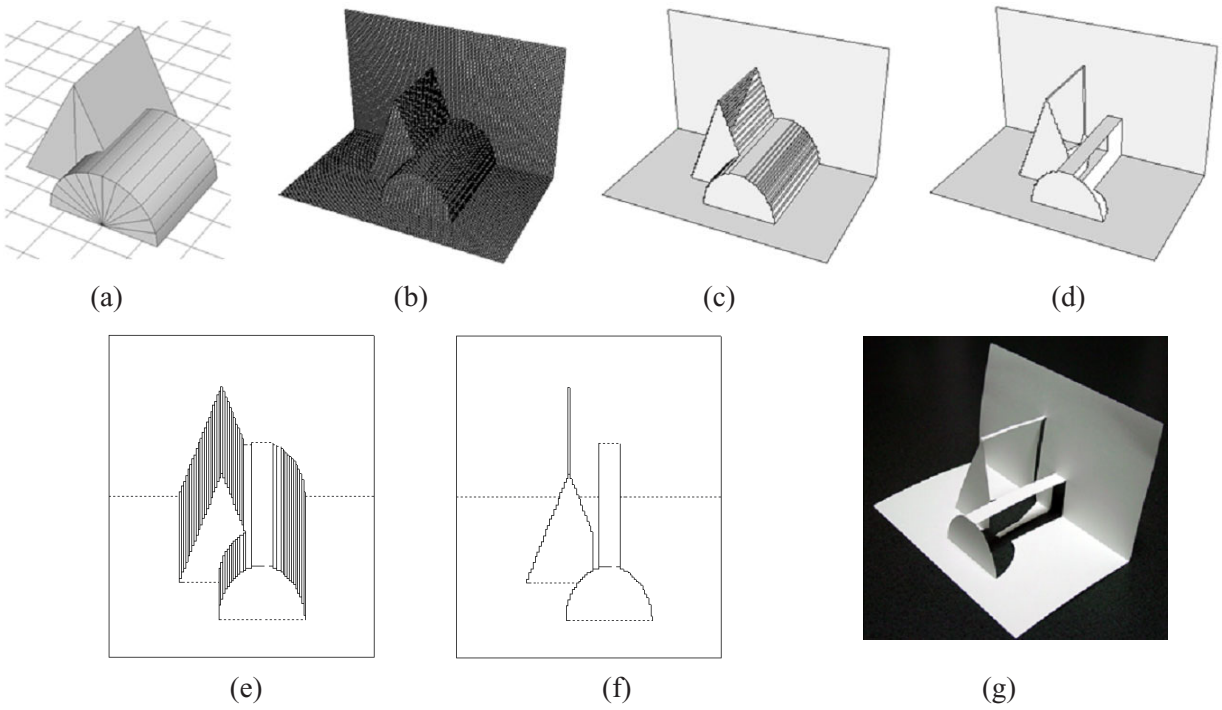


図 5.18: 幅の細い上面の省略

5.4　平面多角形の集合を用いた 90 度型折り紙建築の設計手法

5.2節と 5.3節で述べたボクセル表現を用いた折り紙建築の設計手法では、簡単なデータ構造で形状の表現が可能で、シンプルなアルゴリズムなために実装も容易であった。その反面、ボクセルの性質上、各座標軸に平行な稜線しか作成できないという制約があった。

本節ではこの制約を除くために、平面多角形の集合で折り紙建築を表現する、ボクセル手法とは別の手法を提案する。なお、対象とする折り紙建築の形状は 5.2節と同様の 90 度型のものとする。また、本節で用いる用語の例を図 5.19に示す。

なお、図 5.19の形状は**切り起こし**や座標軸に平行でない稜線が含まれ、 5.2節で提案したデータ構造では実現できなかったが、本節で提案する手法ではこのような形も作成することができる。本節で提案する手法によって作成できる形状は、ボクセル表現の手法で作成できた形状を完全に内包する。

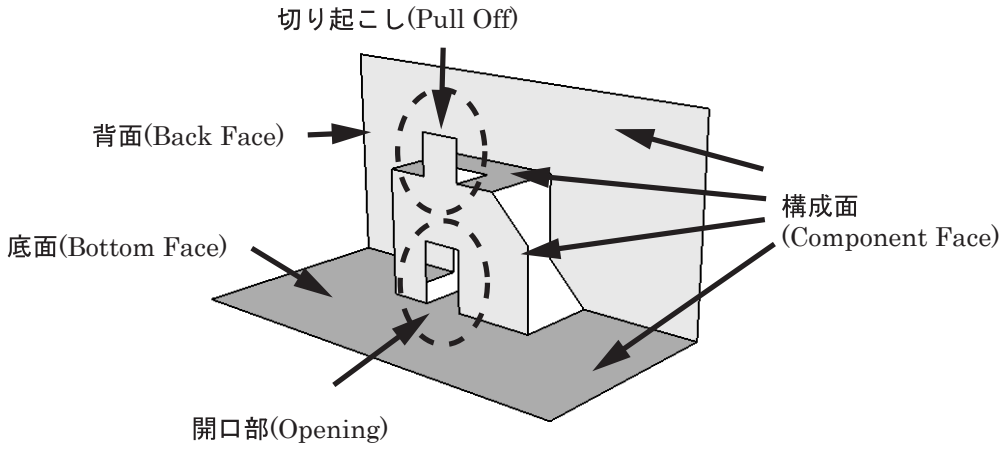


図 5.19: 用語の定義

5.4.1　90 度型折り紙建築の性質

垂直面と水平面

5.2.1節で述べたように本稿で対象とする折り紙建築は、 90 度に開いた状態では、底面に水平な構成面と底面に垂直な構成面の集合から構成される。本節では、それらの構成面をそれぞれ VFace (Vertical Face)、 HFace (Horizontal Face) として区別する（図 5.20）。

カード座標 (3D) と展開図座標 (2D) の関係

対象とする折り紙建築は 1 枚の紙から作成されるため、 180 度開いた状態（展開図）と 90 度に開いて立ち上げた形状には 1 対 1 の対応がある。つまり、 1 つの展開図から立ち上がる立体形状は一意に決まり、またその逆も成り立つ。それぞれの状態において、 3 次元の折り紙建築座標と 2 次元の展開図座標を図 5.21のように底面と VFace の距離を t_v、背面と HFace の距離を t_h と定めると、構成面を成す多角形の各頂点の座標値を次の関係式で相互に変換できる。

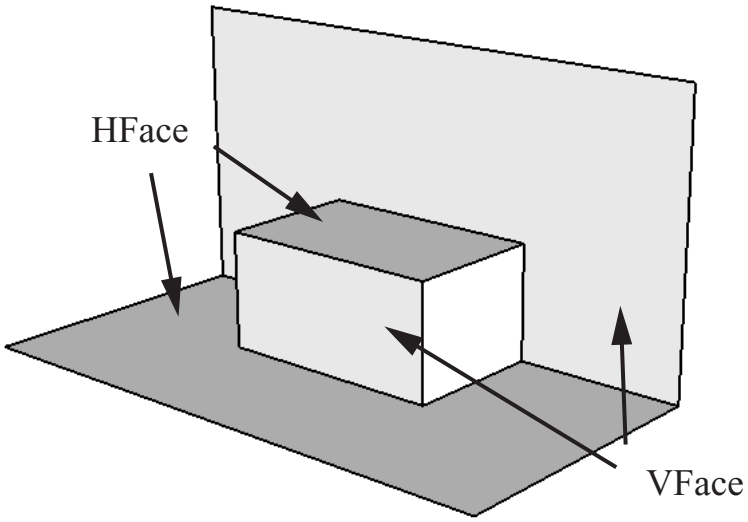


図 5.20: 折り紙建築を構成する VFace と HFace

「折り紙建築座標 (3D) →展開図座標 (2D)」変換

$$\begin{cases} x_{(2D)} = x_{(3D)} \\ y_{(2D)} = z_{(3D)} - y_{(3D)} \end{cases} \tag{5.8}$$

「展開図座標 (2D) →折り紙建築座標 (3D)」変換

VFace の場合:

$$\begin{cases} x_{(3D)} = x_{(2D)} \\ y_{(3D)} = t_v \\ z_{(3D)} = y_{(2D)} + t_v \end{cases} \tag{5.9}$$

HFace の場合:

$$\begin{cases} x_{(3D)} = x_{(2D)} \\ y_{(3D)} = -y_{(2D)} + t_h \\ z_{(3D)} = t_h \end{cases} \tag{5.10}$$

データ構造

計算機内に折り紙建築の形状を保持する場合、保持されたデータから各構成面の輪郭を成す多角形の各頂点の座標値を復元できればよい。前出の式 5.8〜 5.10の関係式から、各頂点の折り紙建築座標（3 次元）での座標値と展開図座標（2 次元）での座標値は相互に変換可能なので、どちらか一方の座標値だけを保持すればよい。次節で述べるが、本手法では展開図座標での各構成面の集合演算を行うため、各頂点の座標値は展開図座標で保持すると都合がよい。こ

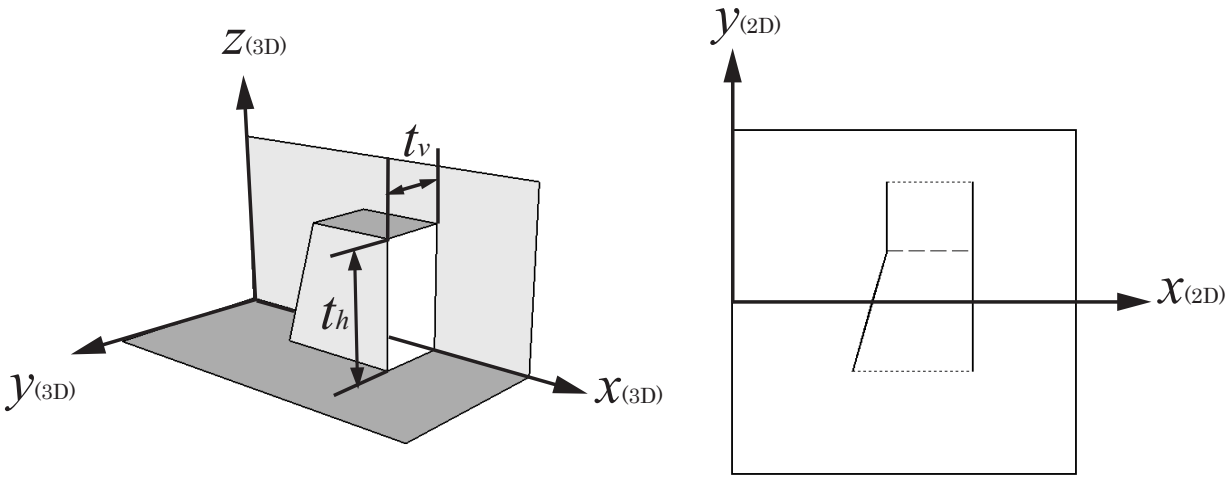


図 5.21: 折り紙建築座標と展開図座標

の場合、 1 つの構成面を表現するのに、構成面の輪郭を成す頂点列の 2 次元座標値と共に、構成面が VFace と HFace のどちらであるかを示すフラグ、及び式 5.9、 5.10中の t の値（VFace の場合は $t = t_v$、 HFace の場合は $t = t_h$）がわかればよい。構成面を表すデータ構造を OAFace とし、それらの集合によって表される折り紙建築のデータ構造を OrigamicArchitecture とすると、これらは図 5.22のような C++ 言語を模した擬似コードで定義できる。

5.4.2 妥当な折り紙建築である条件

展開図作成可能条件

本節で対象とする折り紙建築は 1 枚の紙から作成されるため、各構成面の展開図上の多角形（図 5.22中のOAFace.polygon2D）は互いに重複せず、またそれらの和は工作用紙全体と一致する。これは、図 5.23のように n 個の構成面（底面と背面を含む）から成る折り紙建築の i 番目の構成面の、展開図上の平面多角形を F_i、折り紙建築に用いる工作用紙全体を S とすると、式 5.11のように定義でき、本手法で扱う折り紙建築の形状は常にこの式を満たす必要がある。以降、この条件を**展開図作成可能条件**と呼ぶこととする。

$$
\begin{aligned}
F_i \cap^* F_j &= \phi \quad (i \neq j) \\
F_1 \cup^* F_2 \cup^* \ldots \cup^* F_n &= S
\end{aligned}
\tag{5.11}
$$

式中の $\cap^*$ と $\cup^*$ は、積と和を表す正則化集合演算子 [73] であり、 ϕ は空集合を現す。

立ち上げ可能条件

前節で述べた展開図作成可能条件を満たすことで、 1 枚の紙から作成できることが保証されるが、実際に工作したときに、 90 度に開いたときに形が立ち上がらない場合がある。図 5.24は折り紙建築として適切でないものの例である。

```
enum faceType {VFACE, HFACE};

class Polygon2D {
    list<Point2D> points;
};

class OAFace {
    double value_of_t;
    faceType type_of_this_face;
    Polygon2D polygon_on_unfolded_pattern;
};

class OrigamicArchitecture {
    list<OAFace> faces;
};
```

図 5.22: データ構造の定義

図 5.24(a) には宙に浮いた部分が存在する。 (b) は下方と上方から立ち上がっている部分が分離しているため、台紙を開いた時に形が引き起こされないという問題がある。そこで、台紙を 90 度に開いたときに構成面が立ち上がる条件を**立ち上げ可能条件**と呼び、対象とする折り紙建築はこの条件を満たさなければならないものとする。この条件を満たすか否かを判定するアルゴリズムは 5.4.3節で述べる。

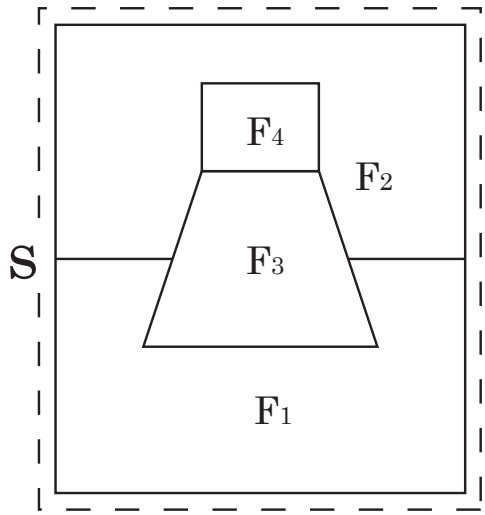


図 5.23: 展開図を構成する平面多角形

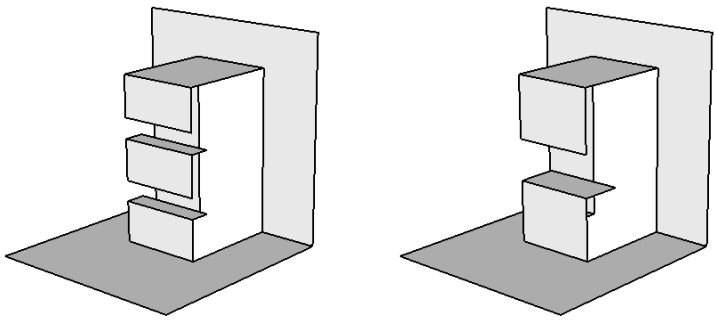


図 5.24: 実現不可能な開口部をもつ場合

5.4.3　計算機による設計支援

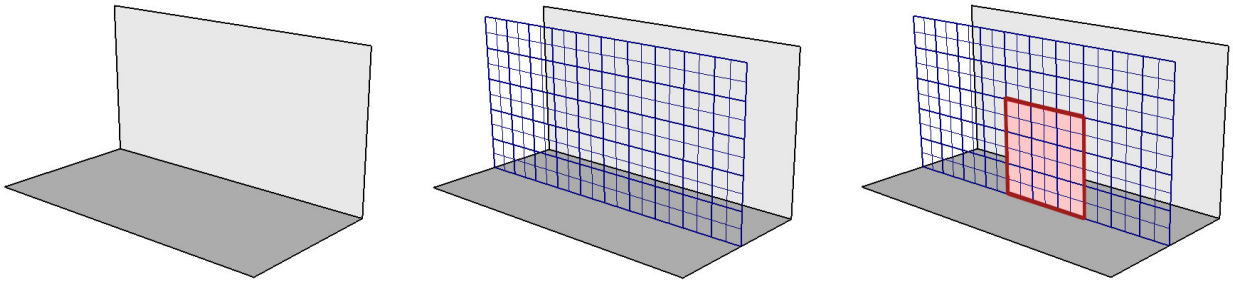
前節では折り紙建築の形状を計算機内に保持するための手法と、形状データが満たすべき条件を示した。本節では、条件を満たした形状をユーザーが容易に作成できるよう、計算機によって折り紙建築の設計を支援する手法を述べる。

ユーザーインターフェース

従来の人の手による折り紙建築の設計は、展開図を直接編集することで行われているため、実際に組み立てを行うまで、 90 度に開いたときに立ち上がる形状を確認できず、試行錯誤に頼らざるをえなかった。そこで本節では、折り紙建築の立体形状を計算機の画面で確認しながら、完成形状を対話的に編集することで、より直感的に設計を行う手法を提案する。折り紙建築は水平または垂直な構成面から成るため、これらを 3 次元空間上の意図した場所に配置できれば折り紙建築の設計を行うことができる。

ここでは、次のようなインターフェースで折り紙建築の設計を行う手法を提案する。

1. 底面と背面のみで構成される折り紙建築を初期状態とする（図 5.25(a)）。
2. 垂直（水平）な基準面の位置を前後（上下）に移動させ、新規に作成する構成面 VFace（HFace）の位置を確定する（図 5.25(b) は垂直な場合を示す）。
3. 2. で決定した基準面上の格子点を選択し、構成面の形（多角形）を入力する（図 5.25(c)）。
4. 構成面を指定された場所に配置し、完成イメージを 3 次元表示する。
5. 2. ～ 4. を繰り返す。



(a)　(b)　(c)

図 5.25: 折り紙建築を設計するインターフェース

新規構成面の追加によるデータ更新

前述のインターフェースで折り紙建築の形状の作成が可能であるが、 5.4.2節で述べた展開図作成可能条件を満たす形になることをユーザーが常に意識しながら作業を進めるのは容易でない。そこで、ユーザーが新しい構成面を追加した時に、展開図作成可能条件を満たすよう

に、システムが自動で全体の形状を修正することとする。これにより、ユーザーは折り紙建築の形状設計に専念できる。具体的には、前項で述べたインターフェースのステップ 3 と 4 の間に、次の処理を行うこととする。

1. ユーザーが新しく追加した 3 次元空間内の構成面について、式 5.8を用いて展開図上での多角形を求め、新しく OAFace を生成する。

2. 新しく生成された OAFace と、既に存在する展開図上の各構成面 OAFace′(i) について、次の多角形の集合演算を行い、既存の構成面の更新を行う。i は各構成面を識別するインデックスである。式中の演算子 ($-^*$) は、正則化集合演算の減算 [73] を意味する。

$$\mathrm{OAFace}'(i).\mathrm{Polygon2D} = \mathrm{OAFace}'(i).\mathrm{Polygon2D} -^* \mathrm{OAFace.Polygon2D} \qquad (5.12)$$

3. OAFaceと全ての OAFace'(i) について、式 5.9, 5.10を用いて展開図座標から 3 次元座標を求め、 3 次元表示を行う。

新規構成面を追加する前の時点で、折り紙建築の形状が展開図作成可能条件を満たしていれば、上記の処理を行った後も、この条件は満たされる。設計の初期状態である、底面と背面のみを持つ折り紙建築は展開図作成可能条件を満たしているため、それ以降に行われる編集操作でも、常に展開図作成可能条件は満たされることになる。この手法では、新規に追加される構成面の形は変更されず、それ以前に作成された構成面の形が修正されるため、ユーザーの直感的な操作を妨げない利点がある。

開口部の作成

構成面の作成に続いて、開口部の作成を行うためのインターフェースを考える。ところで、前述の「多角形の引き算」で既存の面のデータ更新を行う場合、「開口部を作成する操作」には、それと等価な「構成面を作成する操作」が存在する。例えば図 5.26(a) の四角で囲まれた領域に開口部を設け、 (c) のような形状を作成する操作は、 (b) の四角で示される水平な構成面を新規に作成する操作に置き換えることができる。

したがって、開口部を生成するためには、それと等価な構成面の生成を行えばよい。これにより、構成面の作成と開口部の作成を、システム内部では共通のアルゴリズムで行うことができる。

しかし、開口部を生成するために構成面の作成を行うのはユーザーにとって直感的ではない。そこで、この処理はシステム側で行うこととする。つまりユーザーが開口部とする多角形を入力したときには、システムがこれと等価な構成面の生成処理を内部的に実行する。

垂直面と水平面に開口部を生成する命令をそれぞれCreateVHole、CreateHHole、垂直および水平な構成面を生成する命令をそれぞれCreateHFace、CreateVFaceとし、各命令が入力として多角形の輪郭を表す点列 $P[]$ を受け取るとすると、この入力に用いる点列の各座標値を適切に変換することで、次のように開口部の生成命令を構成面の生成命令に変換できる。

$$\mathrm{CreateVHole}(P_v[]) = \mathrm{CreateHFace}(P'_v[])$$
$$\mathrm{CreateHHole}(P_h[]) = \mathrm{CreateVFace}(P'_h[])$$

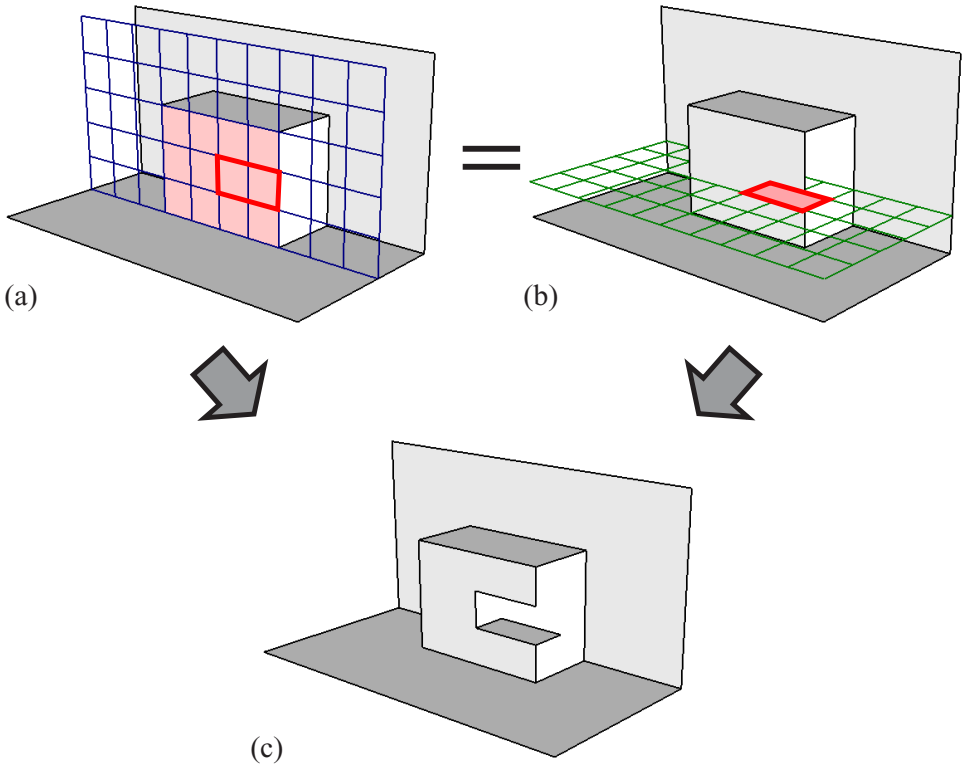


図 5.26: 開口部の作成

ここで、入力された多角形の輪郭を表す点列 $P_v[]$と$P_h[]$ の i 番目の要素の座標値を $(x_v[i], y_v[i], z_v[i])$、$(x_h[i], y_h[i], z_h[i])$ と表した場合、変換後の点列 $P'_v[]$ と $P'_h[]$ の各要素は次のように表現できる。

$$\begin{aligned} P'_v[i] &= (x_v[i], y_v[i] - z_v[i] + minZ, minZ) \\ P'_h[i] &= (x_h[i], minY, z_h[i] - y_h[i] + minY) \end{aligned} \tag{5.13}$$

なお、 $P_v[]$ の各点は底面に垂直な多角形面上にあるため $y_v[i]$ は定数となり、 $P_h[]$ の各点は底面に水平な多角形面上にあるため $z_h[i]$ は定数となる。また、 $minZ$ は点列 $P_v[]$ の z 座標値の最小値、 $minY$ は点列 $P_h[]$ の y 座標値の最小値である。

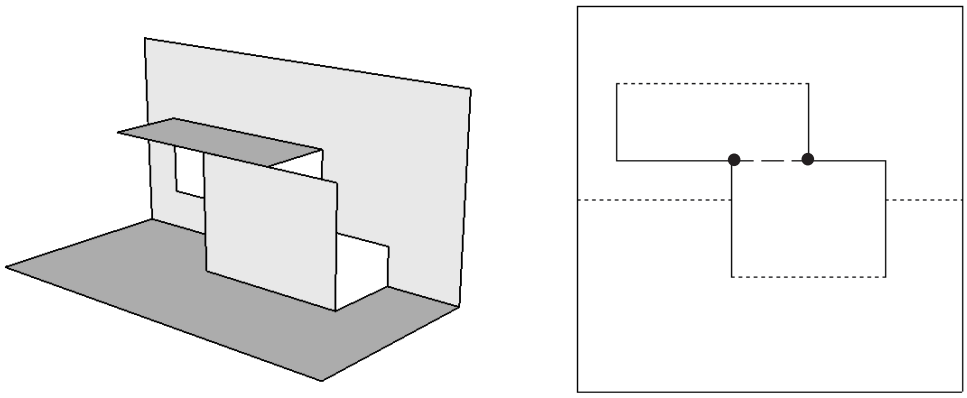
展開図の生成

本手法では、各構成面について展開図座標における位置情報を保持しているため、展開図の生成は各面の輪郭線を展開図座標に基づいて用紙に出力することで容易に行える。しかし、山折り線・谷折り線・切断線によって異なる線種を用いる場合、展開図の出力の前に輪郭線を構成するそれぞれの線分がどのタイプに属するのか判定する必要がある。

折り紙建築は展開図座標および折り紙建築座標における x 軸を中心とした回転で台紙の開閉が行われるため、 x 軸に平行でない線分は折れ線になることはなく、必ず切断線となる。線分が x 軸に平行である場合、その線種は切断線、谷折り線、山折り線のいずれにもなり得る。

この線種の判定は、その線分の一部または全部を共有して展開図上で隣接する構成面との、折り紙建築座標における位置関係から判定する。折り紙建築座標において、互いに線分を共有

しない場合、この線分は切断線であり、凸に共有していれば山折り線、凹に共有していれば谷折り線である。図 5.27(a) のように、隣接する構成面と一部のみを共有する線分の場合には、互いの線分を他方の線分の端点（図 5.27(b) 中の黒丸）で分割を行い、分割された線分それぞれについて上記の判定を行えばよい。



(a) (b)

図 5.27: 互いに一部を共有する線分

CG アニメーションによる開閉シミュレーション

5.2.3節で述べたように各構成面の頂点の折り紙建築座標での座標値 (x, y, z) を、式 5.6で変換することで、折り紙建築の開閉途中の形状を表現できる。図 5.28は 5.4.4節で挙げる例題を、ボクセルによって表現した折り紙建築をアニメーション表示した時と同様に、式 5.6の θ の値を徐々に変化させた CG アニメーションの例である。

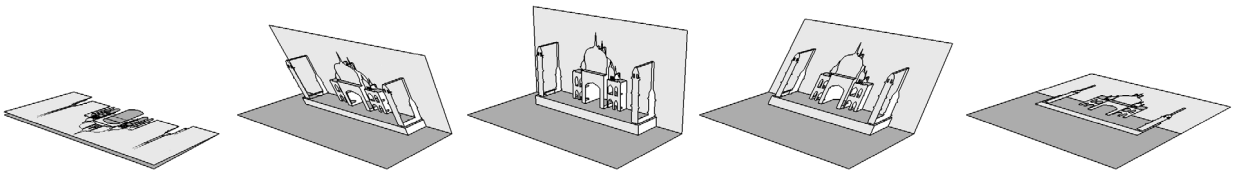


図 5.28: 折り畳み途中の形状表示

立ち上げ可能条件の判定

作成した折り紙建築モデルが、 5.4.2節で述べた立ち上げ可能条件を満たしているか否かを判定するアルゴリズムとして、展開図上の構成面にフラグ立てを行いながら巡回することで妥当性をチェックする手法を提案する。

基本的な考え方は 5.2.3節で提案している妥当性判定の仕組みと同じである。しかし、ボクセルで表現される展開図のような正方形の要素の集合から構成されるわけではないので、隣接する面へ巡回を進めるときには、間に存在する折れ線の種類による判定を行う。

具体的には、展開図を作成後に次のようなアルゴリズムによって立ち上げが可能であるかを判定できる。

1. 展開座標上の各構成面について、以下の方法で巡回を行い、巡回済みのフラグ立てを行う。

   (a) 背面を成す構成面からスタートし、展開図上の山折り線、または谷折り線を介して隣接する構成面を再帰的に巡回する。ただし、構成面のタイプが VFace から HFace に移るときには谷折り線、 HFace から VFace に移るときには山折り線を介していなくてはならない。

   (b) 底面を成す多角形から開始し、展開図上の山折り線、または谷折り線を介して隣接する多角形を再帰的に巡回する。ただし、構成面のタイプが VFace から HFace に移るときには山折り線、 HFace から VFace に移るときには谷折り線を介していなくてはならない。

2. 全ての VFace について 1. の (a) と (b) のフラグのどちらか一方、または両方が立っていないものが存在したら立ち上げ可能条件を満たさず、折り紙建築として適切でない形状である。

上記のアルゴリズムを適用した例を図 5.29に示す。展開図のうち、左側が (1) の巡回を行ったもの、右側が (2) の巡回を行ったものである。 (a)、 (b) は、フラグの立っていない VFace （図中×印）が存在するため、適切でない形状であると判定される。

5.4.4 結果

本手法を PC 上に実装し、実際に折り紙建築の設計を行った。その結果を図 5.30に示す。 (1) は座標軸に平行でない稜線と切り起こし、および開口部を含む例題であり、 (2) はインドのタージマハルを意識して作成したものである。それぞれ、 (a) は本手法で作成した折り紙建築の CG 画像、 (b) は展開図、 (c) は実際に工作を行った写真である。モデルの制作は対話的に行うことができ、 (1) の作品は 2 分程度、 (2) の作品には 30 分程度の設計作業時間を要した。

また、作品に下絵画像を張り付けられるように拡張したものを、一般の大学生 4 名に使用してもらい、その評価を行った。予め下絵を台紙に張ることで、それを参考に形を作ることができる。今回は使用方法をしばらく練習した後、図 5.31に示すような作品をそれぞれ数時間程度の作業で作ることができた。特に折り紙建築の知識が無くとも、本手法を用いることで個人の趣向に応じた様々な形の折り紙建築を容易に作成できることを確認できた。

5.4.5 考察

本章で提案した手法により、ボクセル表現を用いた手法で作成できる折り紙建築よりも自由度の高い作品を対話的な操作で設計できるようになった。折り紙建築の作品を設計する上で、「1 枚の紙から作成される」という制約を満たしながら自由な形を作成することは非常に難しいことであるが、展開図上での論理演算によってこの制約を常に満たすように面の変形を行うことで、制約を意識せずに形の作成に専念できるようになった。面の配置場所の指定と形の指定という単純なユーザーインターフェースを実装することで、折り紙建築の知識のないユー

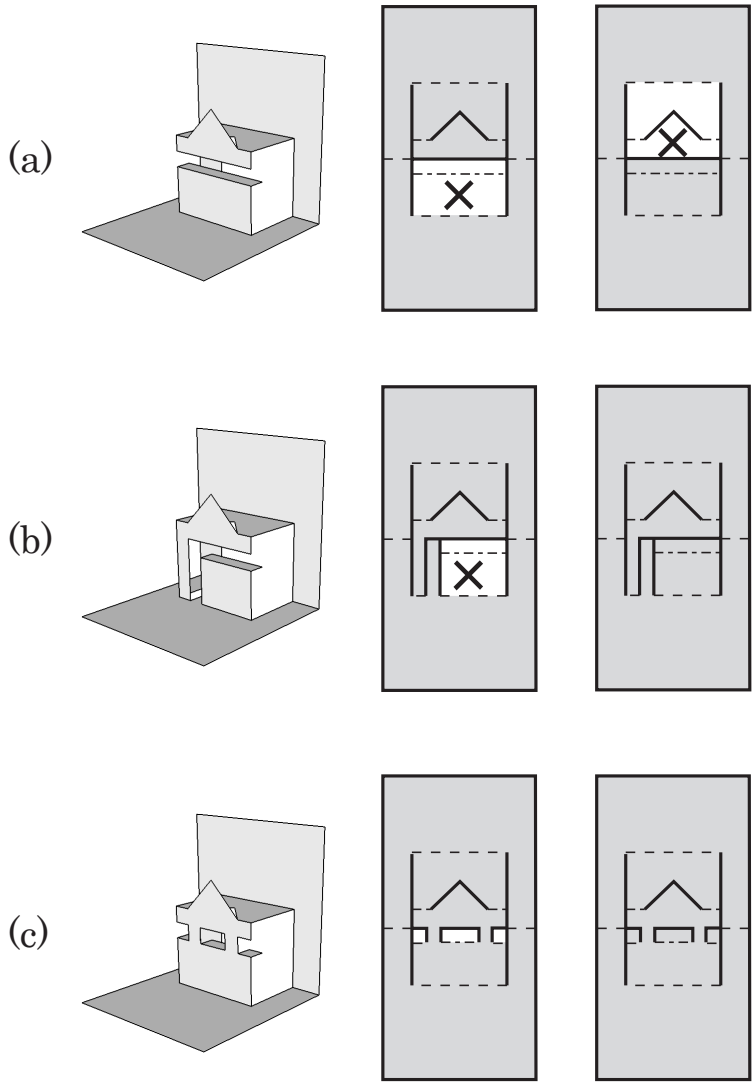


図 5.29: 立ち上がり可能性の判定

ザーでも容易に形を作成できるようになった。しかし、ボクセル表現を用いた手法では、正面と上面が必ず対になって作成されたため、カードを開いたときに形が立ち上がるようにすることは容易であったが、本手法では、正面と上面を別々に入力しなくてはならない点が、やや不便な点に感じられた。折り紙建築では、主に正面の作成がメインの作業になるため、上面は自動的に生成されるようにする工夫があってもよいかもしれない。

また、実際に複数のユーザーに作品を作ってもらうことで、ある箇所の形状を平行移動したり複写する機能があると便利だとの意見が聞かれた。これらの操作を、既存の形状との兼ね合いでどのように実現させるかは今後の課題として残されている。そのほかにも、作品に絵や文字を貼り付けられるようにしたいという要望や、紙の強度を考慮して適切な場所に切り込みを生成するようにしたい、というような要望も挙げられた。本章では、多角形の集合で折り紙建築を作成する手法の基本的な考えを提案したが、ユーザーの立場に立った場合には、さらなる機能の追加も必要であると思われる。そして、これらの機能の中には、さらにアルゴリズムやデータの持ち方の議論が必要なものもあるように感じた。

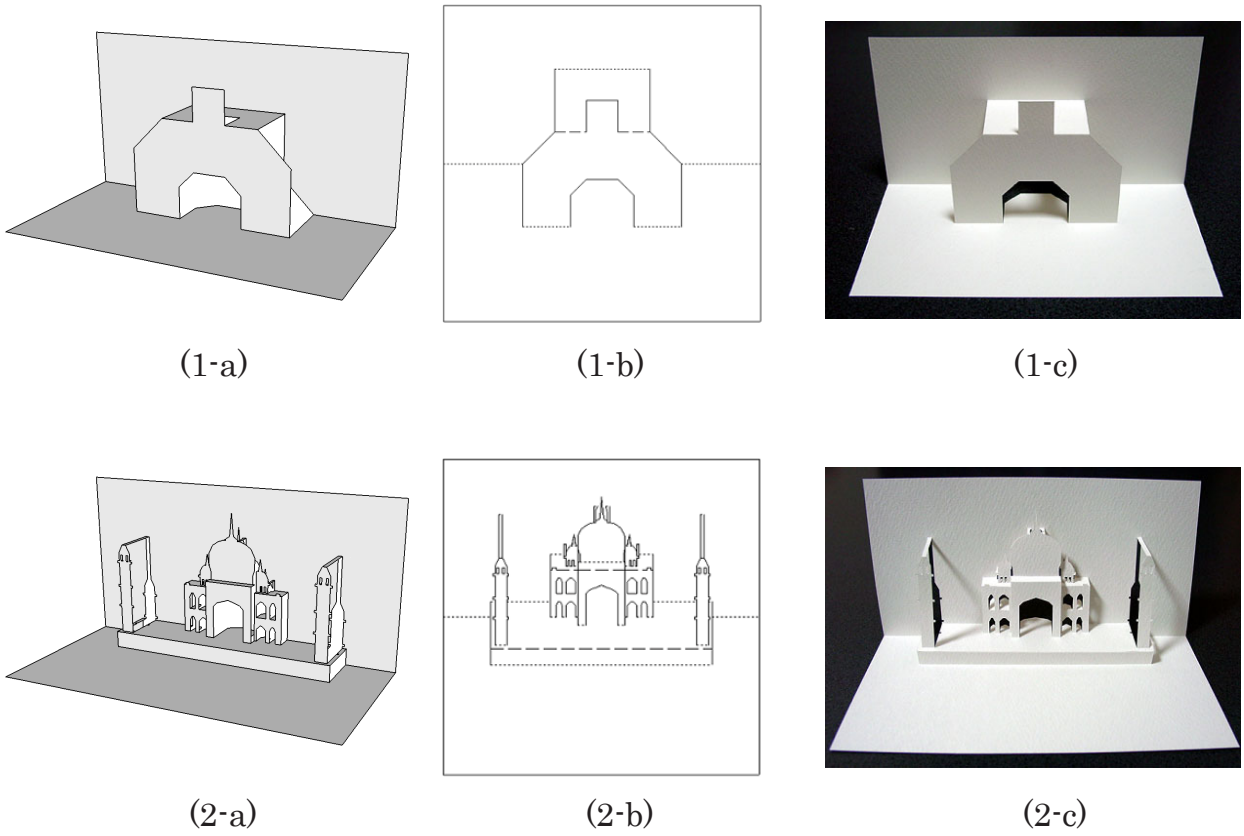


図 5.30: 作品例
(a) CG 画像 (b) 展開図 (c) 作成した折り紙建築

また、さらなる今後の発展として、ボクセル表現を用いた折り紙建築の設計に関する研究のときのように、既存のポリゴンモデルから自動的に生成することを考えてみてもよいかもしれない。ボクセル表現よりも自由度は高いため、より元の形に忠実な折り紙建築を作成可能と思われる。

また、本手法では折れ線によって入力できる多角形面の集合で形状を表現しているため、自由曲線を入力できないという問題がある。制御点を入力することで曲線を生成し、その近似多角形を使用する手法も考えられるが、後から制御点を用いた曲線の修正を行う場合などを考慮するとデータの持たせ方を改善する必要があるであろう。

図 5.31: 大学生の作品

5.5　紙片の格子状組み合わせを用いた 180 度型折り紙建築の設計手法

前節までで、 90 度型の折り紙建築の設計を計算機で支援する手法を提案してきた。本節では、今まで扱ってきた折り紙建築とは大きく異なる図 5.32に示すような、格子状に組み合わさった紙で立体の形を表現する 180 度型の折り紙建築を計算機で設計するための手法を提案する。 180 度型の折り紙建築は、 180 度に開いたときに対象とする形状が立ち上がり、二つ折りで折り畳むことができる。 1 枚の紙では作成できないため、複数のパーツを組み合わせて作る。特にパーツを格子状に組み合わせて形を作る作品は、文献 [74] で多数紹介されている。

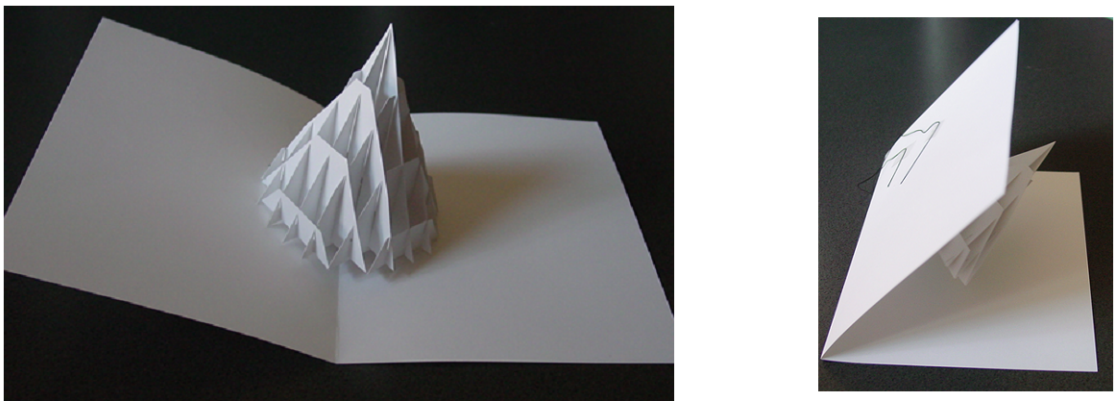


図 5.32: 180 度型折り紙建築の作品例
（左：台紙を開いた様子 右：台紙を閉じている様子）

5.5.1　180 度型折り紙建築の構造

90 度型の折り紙建築は 1 枚の紙で作成できるため、その仕組みを理解しやすいが、 180 度型のものは複数の紙の部品を組み合わせて形を作るため、折り畳みと立ち上がりの仕組みを直感的に理解しにくい。ここでは、この 180 度型の折り紙建築の仕組みをまとめる。

平行リンクと折り畳み

平行リンクは機械部品の一部によく用いられる機構であり、関節を回転させることで、 4 つのリンクで構成される平行四辺形を長方形から直線へ変形させることができる（図 5.33）。また、これらが複数組み合わさった場合でも、各リンクが平行に配置されていれば、同じように変形できる（図 5.34）。

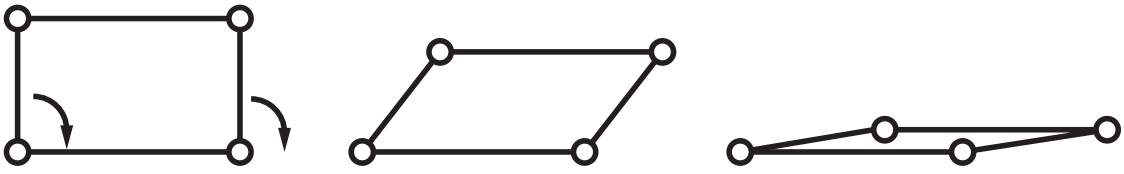


図 5.33: 平行リンクと折り畳み

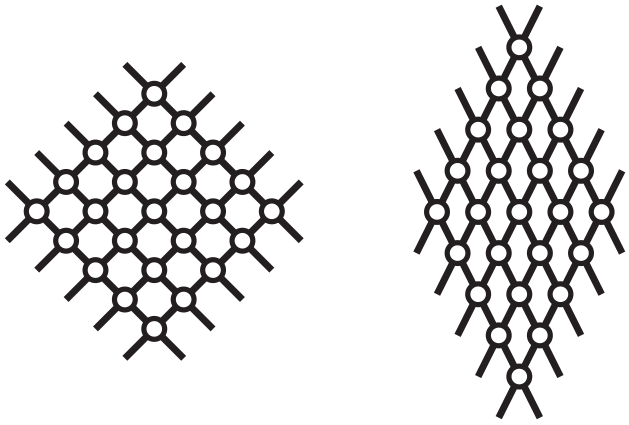


図 5.34: リンク機構の格子状の組み合わせ

紙片に切り込みを入れ、これを図 5.35のように互い違いに組み合わせれば、上方から見たときに、紙を剛体のリンクとみなし、切り込み部を回転の自由度を持つ関節とみなすことで、上記のリンク構造を構成することができる（以降、このように切り込みの入った紙の部品を単に**パーツ**と呼ぶこととする）。従って、このパーツの集合で対象とする立体の形を表現できれば、折り畳みと立ち上がりの変形が可能な立体形状を作成できる。

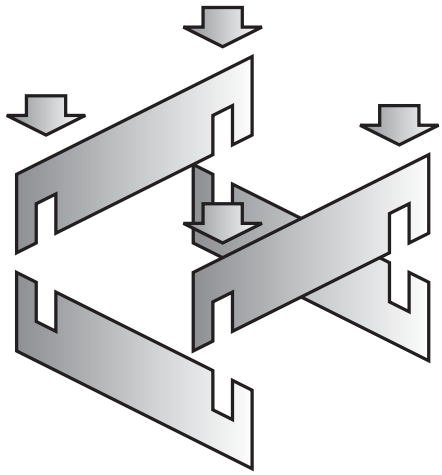


図 5.35: 紙片による平行リンク

台紙への固定

前節で述べたように、互いに組み合わせたパーツによって平行リンクを構成することで、平面へ折り畳める立体を作成できる。 180 度型折り紙建築では、この立体を台紙に固定することで、台紙の開閉に伴って立体の折り畳みと立ち上がりが行われる。

ここで、格子状に構成された立体をどのように台紙に固定すればよいかが問題となる。図 5.36は最も単純な四角柱を台紙に固定し、台紙と共に折り畳んだ写真であり、図 5.37はその様子を模式的に表したものである。この例では、図 5.36の四角柱の奥側に位置する 2 面の底辺

（図 5.37の AB と AD）を台紙に固定すればよい。四角柱の底面を成す点 A, B, D は常に台紙に固定されているが、点 C は台紙が閉じられるにつれ、最初に存在していた辺 E F 上から離れてゆく。この動きにより、四角柱は台紙を閉じると平面へ折り畳まれる。逆に、台紙を開くことで閉じられた四角柱が立ち上がる。なお、図 5.37の AB と AD のように、台紙に固定すべき 2 辺を以降では**固定軸**と呼ぶこととする。

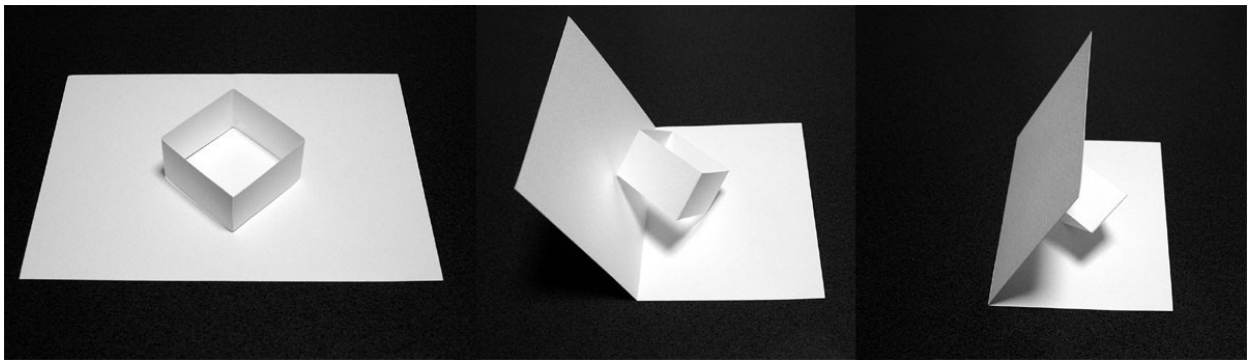


図 5.36: 四角柱の例

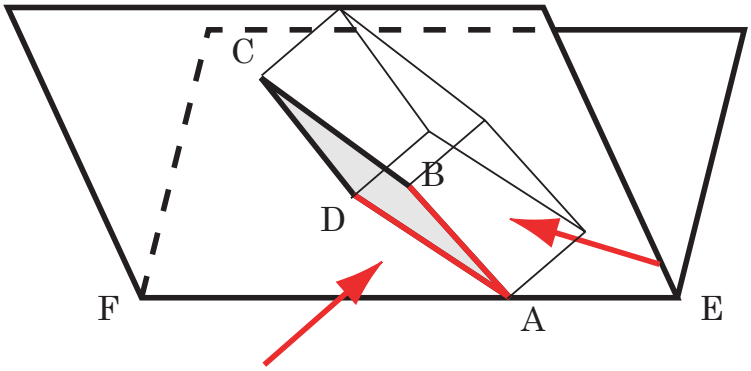


図 5.37: 四角柱の折り畳み

上記の例を参考にすると、図 5.34のように上方から見た時に格子状になっている格子立体は、図 5.38の黒丸で表されるような固定軸上の点を台紙に固定することで、折り畳みと立ち上がりを行えることがわかる。なぜならば、図 5.38の格子によって形成される形状は図 5.37に記したような固定軸で台紙に接する四角柱に内接するからである。なお、格子は平行リンクを構成すればよいため、直交していなくても構わない。また、パーツの間隔は一定でなくてもよい。

格子状に組み合わされた立体が必ずしも図 5.38のような正方形（または平行四辺形）にならない場合は、図 5.39のように最も左側、及び最も上側の点を固定点として選び、その点を通る水平、垂直な線の交点を台紙の折れ線上の点（図 5.37の点 A）とすることで、図 5.37の例と同じように折り畳むことができる。つまりこれは、対象とする形状を内包する四角柱を求め、そのうちの 2 辺を固定軸として選んだことに等しい。固定点は図 5.37の AB, AD 上に少なくとも 1 つずつ存在すれば立体を台紙に固定することができる。

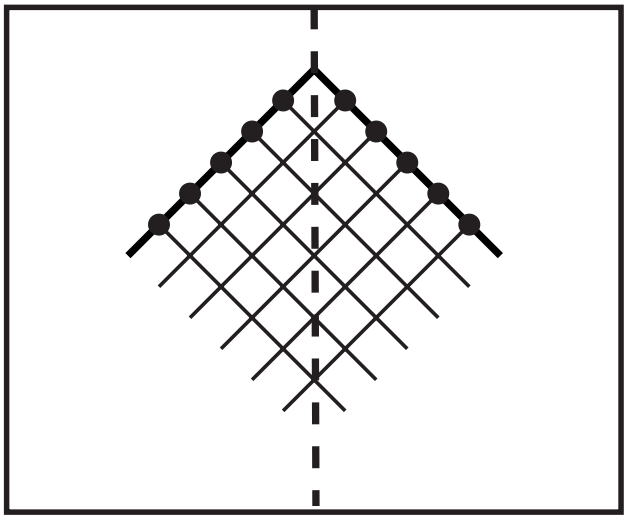


図 5.38: 格子状の立体の固定点

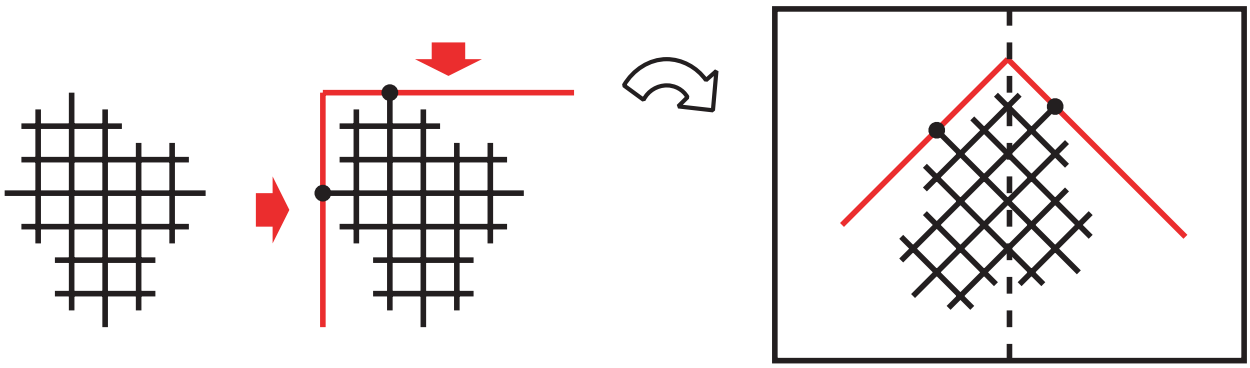


図 5.39: 任意形状に対する固定点の決定

なお、実際に工作するときに、図 5.38,5.39のようにパーツを「点」で台紙に固定するには次のように糸を用いるとよい。

- 台紙上の固定する場所に糸の通る穴を開ける。
- パーツの固定する点に糸をノリなどで固定する。
- パーツに付けた糸を台紙の穴に通し、台紙の裏側からその糸を固定する。

5.5.2　手法の流れ

目的とする形状を 180 度型の折り紙建築で表現するために、各パーツを人の手による試行錯誤で作成するのは容易ではない。そこで本節では、これらの設計を計算機で支援する手法を提案する。

まず、目的の形状を一般の CG ソフトや CAD ソフトでポリゴンモデルとして作成し、その断面を計算することで、形状が立ち上がる折り紙建築の展開図と**配置図**の生成を行う。展開図

は、切り込みを持った各パーツの形状であり、配置図は、そのパーツをどのように台紙に配置すればよいかを表すものである。本手法の流れは以下の通りである。

1. 折り紙建築で表現する形状のポリゴンデータを読み込み表示する（図 5.5.2(a)）。
2. 格子状に組み合わせるパーツの形はポリゴンデータの断面図となる。この断面の位置をユーザーが指定する（図 5.5.2(b)）。
3. 台紙の上下の位置をユーザーが指定する（図 5.5.2(c)）。
4. 指定された断面の位置と台紙の位置から、入力のポリゴンデータを元にパーツの形（断面形状）を算出する（図 5.5.2(d)）。
5. 各パーツに切り込みの生成を行う（図 5.5.2(e)）。
6. 折り紙建築の配置図を生成する（図 5.5.2(f)）。

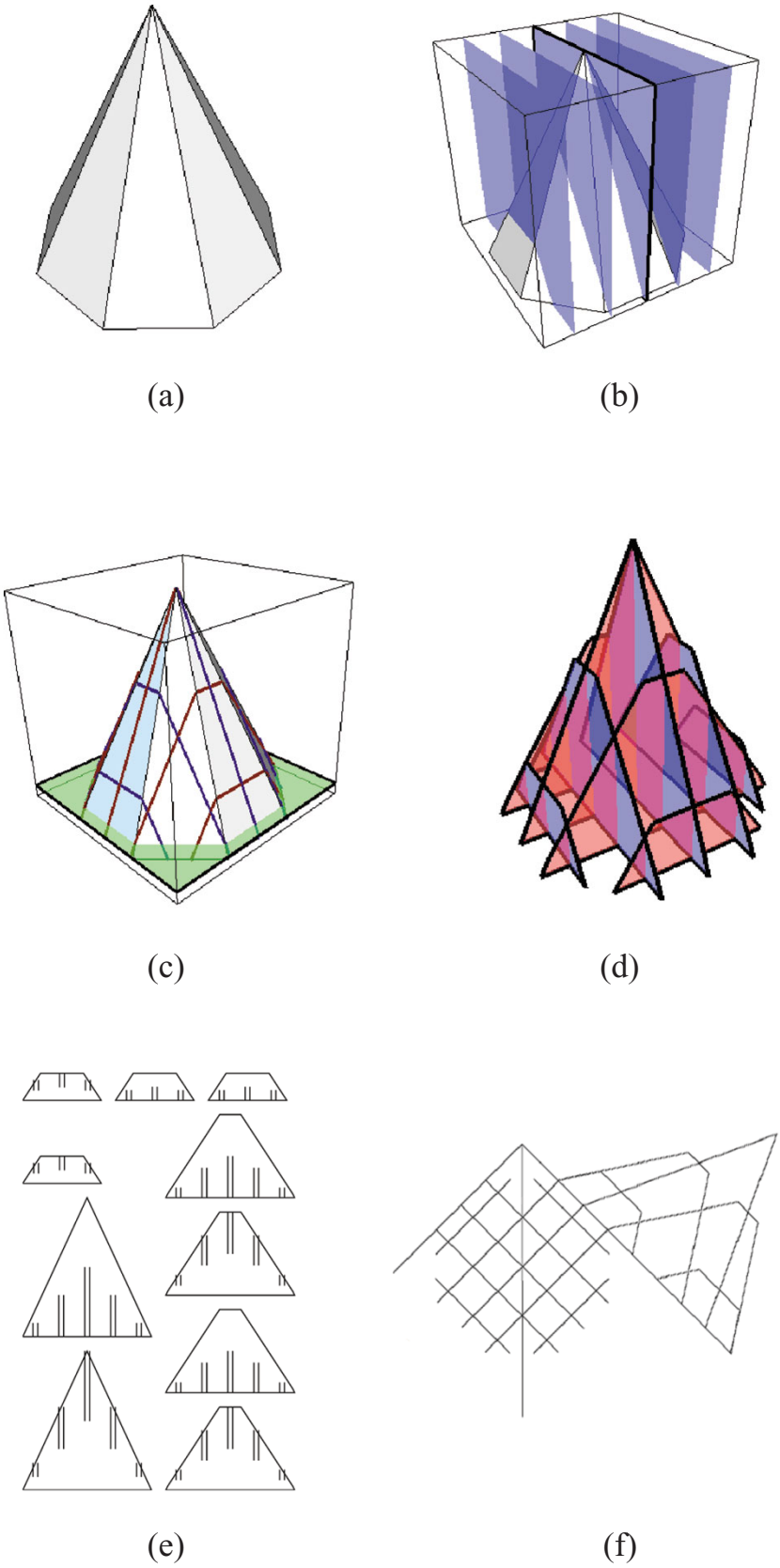


図 5.40: 手法の流れ
(a) ポリゴンデータの読み込み (b) 切断面の指定 (c) 台紙位置の指定
(d) 断面の算出 (e) 切り込みを生成した展開図の出力 (f) 配置図の出力

5.5.3 ポリゴンデータの読み込みと断面の取得

本手法では既存のポリゴンモデルから断面を取得し、それらの組み合わせで折り紙建築を作成する。立体形状の断面を得る必要があるため、対象とする形状は内外の判定ができる 2 多様体である必要がある。ここでは簡単のために球と同位相のものを対象とした。また、データはポリゴンデータの表現手法として一般的に用いられている三角形メッシュであるものとした。

読み込んだポリゴンモデルから断面を取得するために、切断する面の位置をユーザーが指定し、その面によって立体形状を切断する。その際に、互いに平行な複数の断面を同時に取得するものとする。また、簡単のために、本手法では交差する断面は互いに直交するものとした。

断面の計算は、ポリゴンモデルを構成する各面と切断面の交線を求め、端点の一致する交線同士をつなぐことで閉ループを得る。この閉ループが立体形状の断面であり、これを折り紙建築の 1 つのパーツとする。

5.5.4 切り込みの生成

前節で得たパーツについて、それぞれを組み合わせるための「切り込み」を生成する。切り込みはパーツ同士が交わる箇所に生成される。 3 次元空間に配置されたパーツ同士の交線を求めるのはそれほど容易でないが、最初に各パーツが乗る切断面同士の交線を求めることで、切断面の 2 次元空間における、パーツと直線の交わりを求める問題に置き換えることができる（図 5.41左）。パーツ A が乗る平面 P_A と、パーツ B が乗る平面 P_B の交線を L とすると、 L のうち、パーツ A 内に含まれる部分と、パーツ B 内に含まれる部分の共通部分がパーツ A とパーツ B の交線である。

パーツとパーツの交線を求めた後で、その交線上に一方を上方から、他方を下方から切り込みを生成する。切り込みの長さは、図 5.41において $C_A + C_B = V_1V_2$ を満たせばよいが、ここでは簡単のために $C_A = C_B = V_1V_2/2$ とした。切り込みの幅には紙の厚みを考慮して 1, 2 ミリ程度を設けるとスムーズに折り畳みを行える。

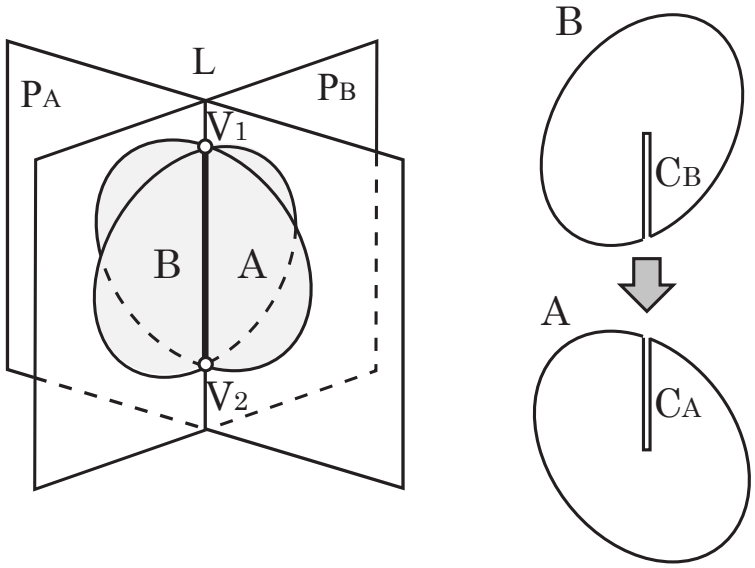


図 5.41: 切り込みの生成

ところで、切り込みによるパーツの組み合わせを行う際に、切り込みの入れ方によっては格子立体がうまく組み立てられない場合がある。例えば図 5.42では、左の組み合わせでは問題無いが、右の組み合わせでは後から差し込むパーツが下に抜け落ちてしまうという問題がある。

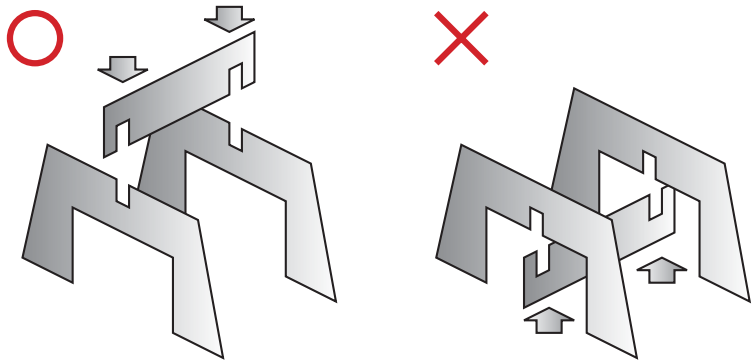


図 5.42: 切り込みの生成方法によって生じる問題

この問題を回避するために、切り込みの生成には以下のアルゴリズムを用い、パーツの抜け落ちが起きないようにした。

1. 次の順に各パーツに 1 から始まる番号を順番に割り当てる
   (a) 固定点が含まれるパーツに番号をつける
   (b) まだ番号のついていないパーツについて、既に番号が割り当てられているパーツと交わる場合は番号を割り当てる
   (c) まだ番号の割り当てられていないパーツが存在しなくなるまで (b) を繰り返す
2. それぞれの交線について、番号の小さいパーツは上側に、番号の大きいパーツには下側に切り込みを入れる。

上記のアルゴリズムにより、固定点の含まれるパーツ（土台となるパーツ）の上に、順番にパーツが組み上げられるようになるため、図 5.42右のような抜け落ちを防ぐことができる。

5.5.5 配置図の生成

折り紙建築を構成するパーツをどのように台紙に配置するかを示したものが配置図である。工作を行うときに、この配置図を参照してパーツの位置決めを行う。本手法では配置図として、図 5.5.2(f) のように 180 度に開いたときにパーツが台紙に接する場所と、台紙を閉じたときにパーツが畳まれる場所、および固定軸を出力するものとする。 180 度に開いたときにパーツが台紙に接する場所は、台紙位置を決定するとき（図 5.5.2(c)）に、台紙平面と切断面の交線を取得すればよい。台紙を閉じたときにパーツが畳まれる場所は次のようにして求めることができる。

図 5.43のように、固定軸を x 軸、 y 軸、台紙に垂直な方向を z 軸とした場合、 x 軸が y 軸に重なるように台紙を折り畳んだときに、点 $P(x, y, z)$ は点 P' へ移動する。ここで、

$$\begin{aligned} P'H' &= PH \\ &= z \end{aligned} \qquad (5.14)$$

$$\begin{aligned} OH' &= OB + BH' \\ &= OB + BH \\ &= y + x \end{aligned}$$

より、 P' の座標は $(-z, x + y, 0)$ となる。

上記の関係式より、 3 次元でのパーツの座標値から配置図上の座標値を求めることができる。これを元に、台紙を閉じたときのパーツの場所を出力する。固定点は、配置図のパーツの輪郭が図 5.43における x 軸または y 軸と交わる点となる。

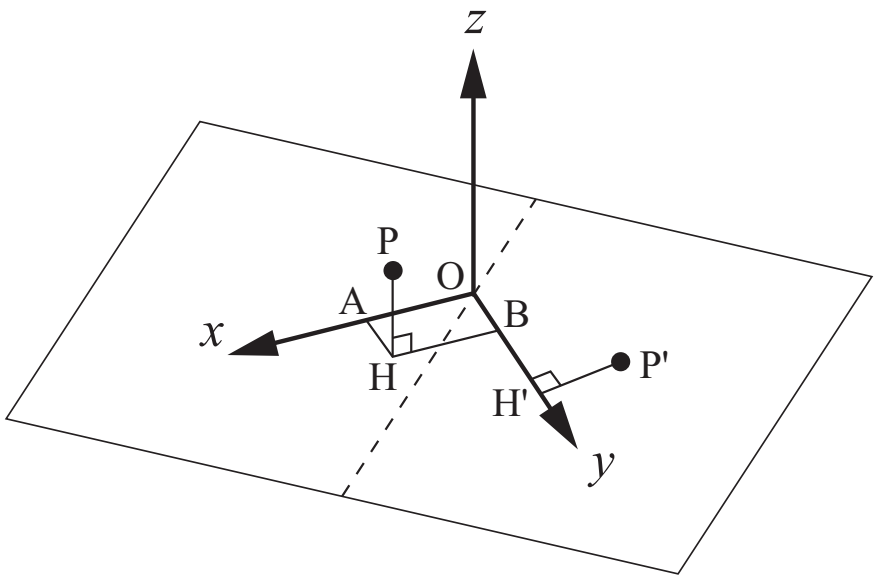


図 5.43: 配置図の生成

5.5.6 結果

本節で提案した手法を PC 上に実装し、実際に折り紙建築を作成した。その結果を図 5.44に示す。 (a) は対象としたネコのポリゴンモデルで面数は 3968 である。 (b) はユーザーが指定した切断面によって生成された断面を CG 表示したものである。切断面は 6 つ指定し、 6 つのパーツから成る。 (c) は配置図、 (d) は切り込みが生成された各パーツの展開図である。組み立て時にわかりやすいように、各パーツの識別番号と切り込みに差し込まれるパーツの番号が出力されている。 (e) と (f) は実際に工作した折り紙建築の写真である。台紙を閉じると、立体形状がスムーズに折り畳まれ、台紙を開くと立体が立ち上がった。切断面の指定と切断の実行は対話的に行うことができた。

5.5.7 考察

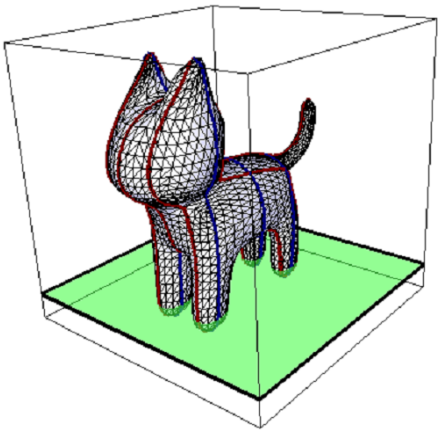
本章では、格子状に組み合わさった紙で立体の形を表現する 180 度型の折り紙建築について、その仕組みをまとめ、 CG の世界でよく用いられるポリゴンモデルからこの折り紙建築の展開図を作成するための手法を提案した。

立体形状の断面図を正確に取得することは、手作業では非常に難しいが、計算機では容易に行うことができる。特に、手作業で断面を求めるのが難しい自由形状（例題のネコのような形）に対して、本手法は有効であると思われる。折り紙建築として、本章で扱ったような 180 度型のものはそれほど一般的に普及しているわけではないが、計算機による支援を行えるようになることで、より容易に様々な形を作れるようになれば、多くの場で使用される可能性があると思われる。

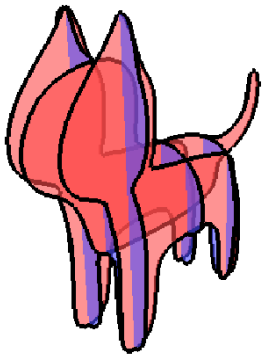
本手法では、ポリゴンモデルに切断面を指定することで、そのパーツの展開図と配置図の生成を行ったが、形状によっては折り畳んだ時に一部がはみ出したり、台紙に干渉して折り畳めないという問題が発生する可能性がある。このような問題点を事前に調べて警告する仕組みもあれば便利であろう。

ところで、一般的なペーパークラフトが、形状の表面によって立体を表現しているのに対し、この 180 度型の折り紙建築はその断面の集合によって立体を表現する点が大きな特徴となっている。折り畳める、という特徴とともに、断面の集合で形を表現する、という特徴を活かせる場が見つかれば、本手法も活用の場が広がると思われる。

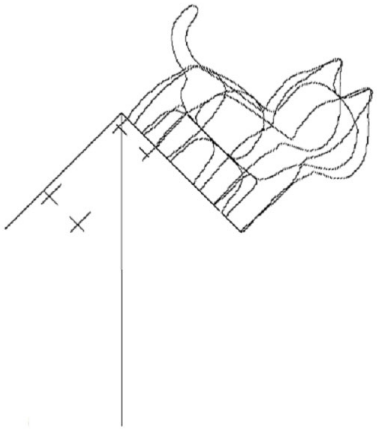
また、本章では断面を格子状に組み合わせる例を扱ったが、 180 度に開いたときに形が立ち上がる例はこれ以外にも、幼児向けの「飛び出す絵本」などにさまざまな事例が存在するので、それを広く扱えるような研究も今後の課題として残されている。



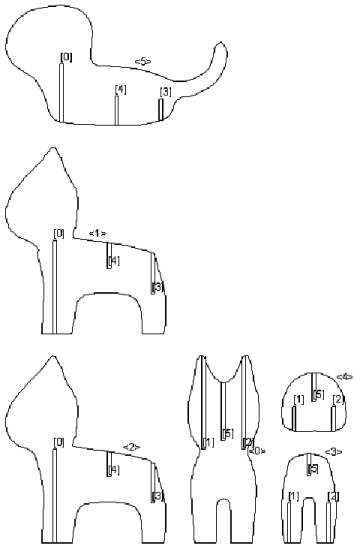
(a)



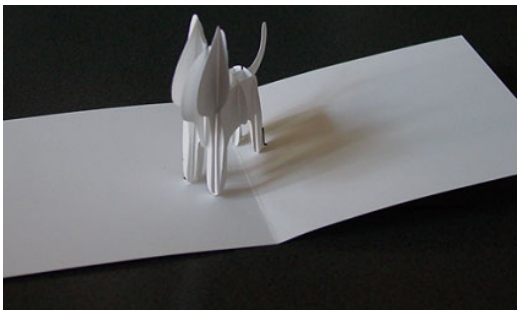
(b)



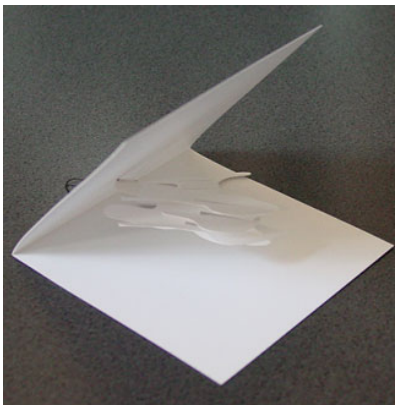
(c)



(d)



(e)



(f)

図 5.44: 180 度型折り紙建築の作成結果
(a) 対象とするポリゴンモデル (b) ユーザが指定した切断面によって生成された断面
(c) 配置図 (d) 各パーツの展開図 (e), (f) 作成した折り紙建築

第 6 章

結論と展望

第 6 章

結論と展望

6.1 結論

本研究では、計算機によって紙模型の設計を支援することを目的とし、ポリゴンモデルおよびメッシュモデルからペーパークラフト用の展開図を生成する手法と、ポップアップカードの一つの技巧として知られている折り紙建築の設計を支援する手法を提案した。

これによって、従来は熟練者による試行錯誤によって設計されていたこれらの展開図を、計算機で容易に作成できるようになった。また、本研究で提案した手法を、実際に一般ユーザーが容易に扱えるようなインターフェースについても考案し、いくつかのアプリケーションに実装して多くの方に使用してもらうことで、その有効性を確認することができた。

各テーマ毎については、以下に述べるような結論を得た。

6.1.1 ポリゴンモデルの展開図作成

CG の世界で一般的に扱われているポリゴンモデルについて、その展開図を作成するためのアルゴリズムをまとめた。また、ペーパークラフトを組み立てるために要するコストについて考察し、展開図の幾何的な値からコストを算出する手法を提案した。その後、実験によってこの評価式の妥当性を検証した。この評価式を基準に、組み立てに要するコストを小さくすることを考慮した展開図の作成法も提案した。これらの手法で得られる展開図を、ペーパークラフトに活用できるように、テクスチャ画像の適応や、のりしろを生成する手法も提案した。また、計算機を用いることの利点を活かすことで、立体と展開図の対応を工作時に確認できたり、立体が展開される様子をアニメーション表示するなどの、工作支援のための手法も提案した。また、これらをアプリケーションとしてユーザーに提供する際に必要となるインターフェースを考案し、それを実際に実装したソフトウェアを開発した。なお、現在ではこのソフトウェアが多くのユーザーに使用され、実用に供されている。これらのユーザーからの意見を元に、今後の課題をまとめた。

6.1.2 メッシュモデルの近似展開図の作成

面の数が非常に多いメッシュモデルはそのまま展開したのでは、工作することが現実的ではないため、 STRIP 形状の集合で近似して展開図を作成する手法を提案した。形状特徴に基づくパーツ分けを行ったのち、 STRIP を生成するための領域分けを行い、新たに特徴線や中心線を切断線に追加する手法を提案した。得られた切断線を平滑化した後、メッシュ簡略化手法

を適用することで、 STRIP による元のモデルの近似形状を得ることができることを示した。これらを展開した展開図を組み立てることで、紙の柔軟性を活かした滑らかな形状を持つ紙模型を作成できることを確かめた。また、この手法では、折り曲げの作業がほとんど発生しないため、工作のコストを小さく抑えられることを確認した。その後、ポリゴンモデルの展開図とのコストの比較と、人の手作業によって作成されたペーパークラフトとの全体的な比較検討を行い、今後の課題をまとめた。

6.1.3 折り紙建築の設計支援

ポップアップカードの技巧で一つである折り紙建築について、 90 度に開いたときに形が立ち上がるものと、 180 度に開いたときに形が立ち上がるものを、計算機で設計するための手法を提案し、以下の結論を得た。

ボクセルを用いた 90 度型折り紙建築の設計支援

90 度型のものについては、折り紙建築の形状を表現するのに、ボクセル表現を用いると効率がよいことを示し、ボクセルで形状を表現したときの、データ保持方法とインターフェース、展開図の作成手法、アニメーション表示などの手法を提案した。これらは開口部を持つ形状も扱うことができる。また、既存のポリゴンモデルから自動的にモデルを作成する方法も提案した。ここで実装したアプリケーションは、教育の場などで実際に多くの方に活用していただいた。それらの結果を踏まえて教育用途への活用などの知見をまとめた。

平面多角形の集合を用いた折り紙建築の設計支援

平面多角形の集合で折り紙建築の形状を表現することで、ボクセル表現を用いた場合には作成できなかった軸に平行でない線分を持つ面や、切り起こしなどの形を含む自由度の高い形状を作成できる手法を提案した。また、新しい構成面を追加するたびに展開平面上での多角形の論理演算を用いることで、妥当な折り紙建築であるための制約をユーザーが意識しなくても済むアルゴリズムを提案した。ボクセルを用いた表現と同様に、対話的に形を作ることができ、展開図の作成や開閉のアニメーションが行えるアプリケーションを開発し、その有効性を確かめた。また、曲線への対応や、望まれる編集機能などを含めた今後の課題をまとめた。

紙片の格子状組み合わせを用いた 180 度型折り紙建築の設計支援

180 度型の折り紙建築の中で、特に紙片を格子状に組み合わせることで形が立ち上がるものについて、その仕組みを明らかにし、ポリゴンモデルの断面形状を取得して計算機で展開図を生成する手法を提案した。配置図の生成方法と、パーツの抜け落ちが発生しない切り込み生成方法などを提案した。これにより、既存のポリゴンモデルの形を表現する 180 度型の折り紙建築の作成を容易に行えるようになった。

6.2 展望

前節でまとめた結論の通り、本研究によってペーパークラフトおよび折り紙建築による紙模型の設計を計算機で支援することが実現した。

面の少ないポリゴンモデルの展開図作成については、ほぼ実用に耐えうるアプリケーションの開発を実現できたと思われる。しかし、面の多いメッシュモデルの展開図作成については、近似精度と組み立てに要するコストとのトレードオフについて今後は評価を行ってゆく必要があると思われる。また、本研究ではメッシュの各面がほぼ均質な大きさのモデルを用いたが、面の細かさが部位によって大きく異なるモデルの扱いも考慮する必要がある。

さらに、本研究で扱ったポリゴンモデル、およびメッシュモデルは、ハーフエッジ構造で表現可能なものであったため、 1 つのエッジを 3 つ以上の面が共有するような非多様体のモデルは含まれなかった。 CG で扱われる形状には、非多様体形状のものも多いため、これらを扱えるように拡張することが望まれる。非多様体のモデルであっても平面多角形の集合であれば、それを平面に展開するのは容易であるが、面と面の位置関係を考慮した「のりしろ」の生成や、簡略化を行う場合の位相の扱いなどに課題がある。現在人の手によって設計されているペーパークラフトには、工作の手間を省くために薄平らな部位（飛行機の翼や動物の耳など）を紙一枚で表現してしまう手法が多く見られるので、これらの技法を用いた展開図の作成も今後の課題である。

折り紙建築については、 1 枚の紙で作成できる 90 度型のものと、格子状に紙片を組み合わせて作成する 180 度型のものに対象を限定したが、一般的なポップアップカードにおいては、これらに限られない様々なバリエーションのものが存在する。今回扱ったタイプに含まれない形状を、どのように扱えるようにしてゆくかは、今後の課題である。

以上で本研究の対象としたペーパークラフトと折り紙建築の設計支援についての個別の展望を述べたが、さらに紙模型の活用という視点で見たときには、パソコンとプリンタ、インターネットが広く普及しつつある現在において、ますます我々の生活に身近になってゆく紙模型の手法を、如何にして我々の生活に役立たせていけるかを提案してゆく必要があるであろう。

例えば、序論で述べたようなホビー産業や工業分野、教育の場など、立体模型の活用が有効な場における、より一層の普及が考えられる。紙模型の展開図の作成が容易になることで、迅速な多品種少量生産が可能となり、例えば個人住宅の模型の作成や、カスタムカーの紙模型、ペットの紙模型など、個人の好みと必要性に応じた形状の作成が可能になるであろう。

また、このような分野に限らず、従来は立体模型が使用されていなかった分野における新しい紙模型の活用方法の考案も望まれる。

謝辞

本研究を進めるにあたり、多くの方々のご協力をいただきました。

鈴木宏正教授には、本研究の指導教官として様々なご指導をいただきました。私が学部生であったころから長きに渡りご指導いただき、研究への取り組みに対する姿勢をお教えいただきました。また、いつでも親身になって相談に乗ってくださり、研究だけでなく生活面においても貴重なご助言を多くいただきました。ここに深く感謝の意を表します。

木村文彦教授には、本研究を進めるにあたり、貴重なご指導をいただきました。ここに感謝の意を表します。

また、毛利尚武教授、山口泰教授、白山晋助教授、五十嵐健夫講師には、本論文の査読をしていただき、貴重なコメントとアドバイスをいただきました。ここに感謝の意を表します。

宮城県仙台市田子中学校の福嶋政一先生には、折り紙建築のソフトウェアを授業でご活用いただき、貴重なご意見をいただきました。

加藤悟助手、碇山みち子技官には、研究室での生活を多くの場面で支えていただきました。

この他、 OB を含む研究室の方々および、陰ながら私の論文作成を応援してくれた家族に深く感謝いたします。

2004 年 2 月 三谷 純

発表論文

投稿論文他（査読付）

1. Jun Mitani, Hiromasa Suzuki, Fumihiko Kimura: “3D Sketch: Sketch-Based Model Reconstruction and Rendering”, Geometric Modeling: Fundamentals and Applications, pp.85-112, 2000.

2. 金井 崇, 鈴木 宏正, 三谷 純, 木村 文彦: “局所的 3 次元モーフィングに基づく 3 角形メッシュの融合演算”, 情報処理学会論文誌, Vol.41, No.3, pp.541-550, 2000.

3. P.A.C. Varley, H. Suzuki, J. Mitani and R.R. Martin: “Interpretation of Single Sketch Input for Mesh and Solid Models”, Int. J. Shape Modeling, Vol. 6, No. 2, pp.207-241, 2001.

4. 三谷純, 鈴木宏正, 宇野弘: “計算機によるボクセルを用いた「折り紙建築」モデルの設計手法”, 情報処理学会誌 Vol.44, No. 5, pp1372-1373, 2003.

5. 三谷純, 鈴木宏正: “立体断面の格子状組み合わせによる 180 度型折り紙建築模型の設計支援”, 日本図学会 Vol.37, No. 3, pp3-8, 2003.

6. 三谷純, 鈴木宏正: “平面多角形の集合による「折り紙建築」モデルの表現と計算機による設計支援”, 情報処理学会論文誌 Vol.45, No. 3, 2004.

講演論文他

1. 金井 崇, 三谷 純, 鈴木 宏正, 木村 文彦: “3 次元形状モーフィングに基づく三角形メッシュの融合演算”, 1998 年度精密工学会秋季大会学術講演会講演論文集, p.90, 1998.

2. 三谷純, 鈴木宏正, 木村文彦: “3 次元ポリゴンモデルの展開図作成”, グラフィックスと CAD 研究会 情報処理学会研究報告 1999-CG-096, pp. 13-18, 1999.

3. Takashi Kanai, Hiromasa Suzuki, Jun Mitani and Fumihiko Kimura: “Interactive Mesh Fusion Based on Local 3D Metamorphosis”, Proc. Graphics Interface ’99, pp.148-156, 1999.

4. 三谷純, 鈴木宏正, 木村文彦: “3 次元スケッチ: スケッチ手法による意匠設計支援のためのモデル構築とレンダリング”, 2001 年度精密工学会秋季大会学術講演会講演論文集, 2001.

5. 三谷純, 鈴木宏正, 木村文彦, 宇野弘: “ボクセルを用いた「折り紙建築」形状の設計”, 2002 年度精密工学会春季大会学術講演会講演論文集, p.265, 2002.

6. 三谷純, 鈴木宏正, 宇野弘: “3D ポリゴンモデルからの「折り紙建築」モデル生成手法”, グラフィックスと CAD 研究会 情報処理学会研究報告 2002-CG-107, pp.7-12, 2002.

7. 鈴木宏正, 三谷純: “3 次元 CG と紙工作を融合した図工教材システムの開発”, 平成 14 年度松下視聴覚教育助成成果報告集, pp.149-158, 2002.

8. 三谷純, 鈴木宏正, 宇野弘: “計算機による「折り紙建築」形状の設計支援”, 2002 年度日本図学会大会学術講演論文集, pp.35-40, 2002.

9. 高橋祐輝, 三谷純, 鈴木宏正: “展開図を利用した 3 次元ペイントシステム”, 2002 年度日本図学会本部例会講演論文集, 2002.

10. 三谷純, 鈴木宏正, 宇野弘: “計算機による展開図の作成とその応用”, 2002 年度日本図学会図学教育教育研究会講演論文集, 2002.

11. 三谷純, 鈴木宏正, 宇野弘: “計算機による 180 度系折り紙建築模型の設計支援”, 2002 年度日本図学会本部例会講演論文集, 2002.

12. 三谷純, 鈴木宏正: “格子状の組合せを用いた 180 度型折り紙建築模型の設計支援”, 2003 年度精密工学会春季大会学術講演会講演論文集, 2003.

13. 三谷純, 鈴木宏正: “90 度型折り紙建築模型の設計支援”, 2003 年度精密工学会春季大会学術講演会講演論文集, 2003.

14. 三谷純, 鈴木宏正: “多角形面の集合による表現を用いた折り紙建築模型の設計支援”, 2003 年度日本図学会大会講演論文集, 2003.

15. 三谷純, 鈴木宏正: “紙模型のためのメッシュモデルの近似展開図生成手法”, 2003 年度精密工学会秋季大会学術講演会講演論文集, 2003.

16. 三谷純, 鈴木宏正: “ポリゴンモデルの展開図組み立てに要するコスト評価法”, 2003 年度日本図学会本部例会学術講演論文集, p27-32, 2003.

参考文献

[1] 前田修治. 数式・係数による展開法. 前田板金展開研究所, 2003.

[2] 学校図書. 中学校数学 3. 学校図書, 2002.

[3] 九州工業大学 情報工学部 機械システム工学科 ラピッドタイピング.
http://designer.mse.kyutech.ac.jp/techInfo/rpindex-j.html.

[4] H. Kodama, S. Igarashi, and K. Saito. A new stereolithography process for improved performance and surface roughness, rapid product development. pp. 93–102.

[5] H. Kodama. Automatic method for fabricating a three-dimensional plastic model with photo-hardening polymer. *Rev.Sci.Instrum.*, Vol. 52, No. 11, p. 1770, 1981.

[6] 桜井勇亮, 鈴木宏正, 金井崇, 木村文彦. 頂点オフセットによる三角形メッシュのシェリング - 光造形データ作成への応用 -. 精密工学会誌, Vol. 64, No. 6, pp. 835–839, 1998.

[7] ローランド ディー. ジー. 株式会社.
http://www.rolanddg.co.jp/.

[8] スルッと KANSAI 協議会. 電車・バスペーパークラフト本. 阪急電鉄, 2001.

[9] 幕張と建築のアートガイド.
http://www.anjo.gr.jp/makuhari/index.html.

[10] 手づくりタウン・ペーパークラフト・折り紙.
http://www.tezukuritown.com/papercraft/papercraft_top.htm.

[11] 日本折り紙学会オフィシャルホームページ 折り紙探偵団.
http://www.origami.gr.jp/.

[12] 宮崎慎也, 安田孝美, 横井茂樹, 鳥脇純一郎. 仮想空間における折り紙の対話型操作の実現. 情報処理学会論文誌, Vol. 34, No. 9, pp. 1994–2001, 1993.

[13] Kei Craft.
http://www.keicraft.com/.

[14] ジャストシステム出版部. デジタルペーパークラフト. ジャストシステム, 2001.

[15] ペーパークラフトデータベース.
http://www.j-park.net/papercraft/bookmark/list.jsp.

[16] グラフテック株式会社.
http://www.graphtec.co.jp/.

[17] Takahiro Yonemura and Sadahiko Nagae. Design procedure on a newly developed paper craft. *Journal for Geometry and Graphics*, pp. 99–107.

[18] 和田弘重, 高井那美, 高井昌彰. 組立ての容易さと曲げ変形を考慮した 3d ポリゴンモデルの展開図生成. 情報処理学会研究報告, 第 2003 巻, pp. 45–50. 情報処理学会, 2003.

[19] Gershon Elber. Model fabrication using surface layout projection. *Computer-aided Design*, Vol. 27, No. 4, pp. 283–291, 1995.

[20] SIGGRAPH 98 Teapot Assembly.
http://www.siggraph.org/s98/conference/teapot/.

[21] 茶谷正洋. 折り紙建築虎の巻. 彰国社, 1985.

[22] Y.T.LEE, S.B.TOR, and E.L.SOO. Mathematical modelling and simulation of pop-up books. *Computers & Graphics*, Vol. 20, No. 1, pp. 21–31, 1996.

[23] Andrew Glassner. Andrew glassner's notebook. *IEEE Computer Graphics and Applications*, Vol. 22, No. 1, pp. 79–86, 2002.

[24] Andrew Glassner. Andrew glassner's notebook. *IEEE Computer Graphics and Applications*, Vol. 22, No. 2, pp. 74–85, 2002.

[25] 茶谷正洋. 折り紙建築型紙集. 彰国社, 1984.

[26] 茶谷正洋. 折り紙建築型紙集 -2. 彰国社, 1986.

[27] 茶谷正洋, 中村倭文夫, 安藤直見. 実例パソコン折り紙建築と折り紙. 講談社, 1987.

[28] 磯田浩, 鈴木賢次郎. 図学入門. 東京大学出版会, 1986.

[29] 図形科学のオンラインテキスト.
http://lecture.ecc.u-tokyo.ac.jp/~okatu/zugaku/4_1line_surface.html.

[30] H. Pottmann and G. Farin. Developable rational bèzier and b-spline surfaces. *Comput. Aided Geom.*, Vol. 12, No. 5, pp. 513–531, 1995.

[31] Josef Hoschek. Approximation of surfaces of revolution by developable surfaces. *Computer-aided Design*, Vol. 30, No. 10, pp. 757–763, 1998.

[32] Simon Kolmanic and Nikola GUid, editors. *The Flattening of Arbitrary Surfaces by Approximation with Developable Stripes*, Vol. 208 of *IFIP Conference Proceedings*. Kluwer, 2001.

[33] Martti Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland, 1988.

[34] 乾正知. 整合的な公差に基づく多角形の頑健な集合演算. 情報処理学会論文誌, Vol. 37, No. 6, pp. 1099–1106, 1996.

[35] 竹居智久, 木村知史. 製造部門が導く作りやすい製品設計. 日経デジタルエンジニアリング, No. 11, pp. 110–115, 2000.

[36] 組立性評価システム.
http://www.englink21.com/i-eng/guest/10pl0003.htm.

[37] ヒューマンインタフェース 101 冊の本.
http://web.sfc.keio.ac.jp/~yasumura/hi_ref.html.

[38] AutoCAD.
http://www.autocad.com/.

[39] 株式会社 終作.
http://www.shusaku.co.jp/www/.

[40] Metasequoia.
http://www21.ocn.ne.jp/~mizno/metaseq/index.html.

[41] 3D Studio MAX.
http://www.ktx.com/.

[42] Victor Milenkovic. Rotational polygon containment and minimum enclosure using only robust 2d constructions. *Computational Geometry*, Vol. 13, No. 1, pp. 3–19, 1999.

[43] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. Rectangle-packing-based module placement. In *International Conference on Computer Aided Design*, pp. 472–479, 1995.

[44] エービーネット株式会社.
http://www.abnet.ne.jp/.

[45] ペパクラデザイナー.
http://www.e-cardmodel.com/pepakura-ja/index.html.

[46] デザインファクトリー, 三谷純. パソコンで作るペーパークラフト！ ペパクラデザイナー徹底ガイド. ソシム, 2003.

[47] ホームページ 鬼屋.
http://prara.infoseek.livedoor.net/index2.htm.

[48] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of the conference on Visualization '02*, pp. 355–362, 2002.

[49] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jerome Maillot. Least squares conformal maps for automatic texture atlas generation. In *Computer Graphics (Proc. SIGGRAPH 02)*, pp. 362–371.

[50] Alla Sheffer. Spanning tree seams for reducing parameterization distortion of triangulated surfaces.

[51] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In Eugene Fiume, editor, *Computer Graphics (Proc. SIGGRAPH 01)*, pp. 409–416. ACM Press / ACM SIGGRAPH, 2001.

[52] Jonathan D. Cohen. Concepts and algorithms for polygonal simplification. In *SIGGRAPH'99 course notes No.20 Interactive Walkthroughs of Large Geometric Datasets*, pp. C1–C34, 1999.

[53] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, Vol. 22, No. 1, pp. 37–54, 1998.

[54] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *Computer Graphics*, Vol. 31, No. Annual Conference Series, pp. 209–216, 1997.

[55] QSlim Simplification Software. `http://graphics.cs.uiuc.edu/~garland/software/qslim.html`.

[56] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide*. Addison-Wesley, 1993.

[57] Michael Garland, Andrew Willmott, and Paul S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Computer Graphics (Proc. SIGGRAPH 01)*, pp. 49–58.

[58] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *Computer Graphics (Proc. SIGGRAPH 95)*, pp. 173–182.

[59] M. Djebali, M. Melkemi, and N. Sapidis. Range-image segmentation and model reconstruction based on a fit-and-merge strategy. In *Computer Graphics (Proc. SIGGRAPH 02)*, pp. 127–138. ACM Press / ACM SIGGRAPH, 2002.

[60] Emanuele Trucco and Robert B. Fisher. Experiments in curvature-based segmentation of range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 2, pp. 177–182, 1995.

[61] S. Katz and L. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, Vol. 22, No. 3, pp. 954–961, 2003.

[62] A. Hubeli and M. Gross. Multiresolution features extraction from unstructured meshes. In *In Proc. of IEEE Visualization Conf.*, pp. 287–294, 2001.

[63] Gabriel Taubin. A signal processing approach to fair surface design. In *Computer Graphics (Proc. SIGGRAPH 95)*, pp. 351–358, 1995.

[64] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics*, Vol. 26, No. 2, pp. 55–64, 1992.

[65] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. *Computer Graphics*, Vol. 27, No. Annual Conference Series, pp. 19–26, 1993.

[66] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, Vol. 22, No. 3, pp. 485–493, 2003.

[67] 中沢圭子. 折り紙建築グリーティングカード集. 彰国社, 1994.

[68] 茶谷正洋, 中沢圭子. 折り紙建築京の旅. 彰国社, 1994.

[69] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *IEEE Computer*, Vol. 26, No. 7, pp. 51–64, 1993.

[70] Hongshen Chen and Shiaofan Fang. Fast voxelization of three-dimensional sythetic objects. *Graphics Tools*, Vol. 3, No. 4, pp. 33–45, 1998.

[71] OpenGL Architecture Review Board. OpenGL リファレンスマニュアル第 2 版. ピアソン・エデュケーション, 1999.

[72] 間瀬茂, 上田修功. モルフォロジーと画像解析. 電子情報通信学会誌, Vol. 74, No. 2, pp. 166–174, 1991.

[73] A.A.G. Requicha. Representation of rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, Vol. 12, No. 4, pp. 437–464, 1980.

[74] 茶谷正洋, 中沢圭子. とびだすペーパークラフト・ 7 マジックハウス. 雄鶏社, 1994.