折りたたみ可能な三次元物体に対する3Dプリンタによる造形手法

A Method for Creating Foldable Three Dimensional Objects Using 3D Printer

情報工学専攻 島津 貴行
SHIMAZU Takayuki

概 要

家具のような板状の部品からなる三次元形状を，収納スペースを節約するために，折りたたむことが出来るように変形する手法が Li らによって提案されている．この手法は，形状と折りたたみ方向を入力し，最適化問題を解くことによって，折りたためる形状を求めている．本論文では，各部品を個別に造形しても組み立てられるように分解する手法を与える．3D プリンタで造形する際には，部品の配置を多角形の平面詰め込み問題の解法を用いて求め，サポート材の使用量の削減や造形時間の短縮を図った．また，実際に造形物を作成し，組み立て可能であることを確認した．また，既存手法を基に収納したい領域を指定し，なるべくその領域内に収納できるような折りたたみ形状を求める方法を与える．

キーワード： 最適化問題，平面詰込み問題，3D プリンタ

1 序論

私たちの生活の中で，空間を効率良く利用するために折りたたみ傘，パイプ椅子のような折りたためる物体というものは数多く存在する．このような折りたたみ可能な物体は，デザイナーのアイデアと多くの実験によって生まれることが普通であり，誰でも折りたたみ可能なものが作れるわけでは無い．そこで本研究では，家具のような板状の部品からなる三次元物体を入力とし，自動で折りたたみ可能な形状を求め，実際に 3D プリンタで折りたたみ可能な物体を造形する．また，3D プリンタで造形する際には，無駄な材料の使用量の削減や造形にかかる時間の短縮を行い，効率的に造形ができる手法を提案する．

今日，物体の変形や折りたたみについての研究は広く行われている．飛び出す絵本のようなポップアップと呼ばれる，物体を表す紙を折りたたむことができるようにする Li らの研究 [3] や，三次元物体をつながった状態で分割し，それを変形させると立方体になる Zhou らの研究 [4] がある．防災用の折りたためるヘルメットなど，日常の中の問題にも物体の折りたたみに対する研究は関わりを持つ．

また，近年，3D プリンタは操作性や精度の高さから一般の人でも利用することが増えてきている．本研究では，FDM(Fused Deposition Modeling/熱溶解積層方式) 3D プリンタを利用する．

既存研究 [2] では，三次元物体と折りたたみ方向を入力として，入れ子型の最適化問題を解くことによって折りたたみ可能な形状を求めている．この入れ子型の最適化問題は，二つの段階から成る．内部ループでは，重み付き最大独立集合問題の解を用いて最適な折りたたみ可能な形状を決定し，外部ループでは，最適な折りたたみ順序を決める．

本研究では，既存手法の最適化問題において，ボクセル化を用いて，折りたたんだ後の領域に関するコストを追加することで，なるべくその領域内に収まる形状を求める．また，この結果から得られた形状を 3D プリンタで造形する際には，形状を部品に分解し，部品ごとに造形しても組み立てられる手法を提案する．[1] の bottom-left 法を用いた解法を基にした平面詰め込み問題の解から，部品の配置を決定し造形する．

2 既存研究

既存研究 [2] では，直方体で近似できる領域のメッシュの集合であるメッシュデータ O と一つの折りたたみ方向 $\vec{d}$ を入力として，物体がその方向に対して垂直に折りたたむことができるような形状を求める．また，折りたたみ可能な形状を得るための物体 O に対する操作は，蝶つがいの挿入と部分的な縮小のみを許す．蝶つがいを以後，ヒンジと呼ぶことにする．

既存手法 [2] の手順を表したものが図 1 である．[2] では，扱う形状の次元を下げる簡略化と，問題を小さくする分割をし，最適化問題に定式化することで折りたたみ可能な形状を得る．まず，入力の三次元形状 O が与えられたとき，入力形状を構造 $\mathcal{I}$ に簡略化する (図 1(b))．次に，この構造 $\mathcal{I}$ を依存構造集合に分割する (図 1(c))．最後に，構造に対する操作を決定し，折りたたみ可能な形状を出力する (図 1(d))．

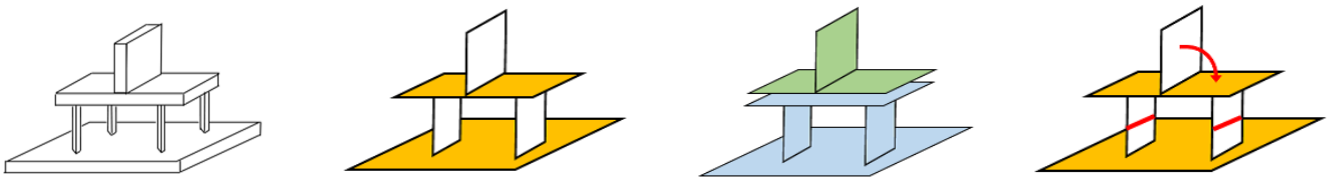
2.1 定義と問題提起

入力物体を簡略化したものから折りたたみ解を得るために用いる定義と，最適な折りたたみ可能な解の定式化を行う．

構造と依存構造集合

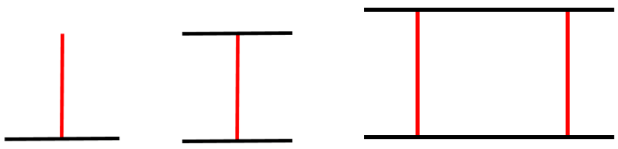
簡略化した構造 $\mathcal{I}$ である長方形をパッチと呼ぶことにする．また，$\vec{d}$ に垂直であるパッチを固定パッチ，それ以外のパッチを可変パッチと呼ぶ．最小の構造を，単位構造と呼ぶ．これには二種類あり，一つは，二つの固定パッチと一つの可変パッチから成り，もう一つは，固定パッチと可変パッチ，一つずつから成る．前者を H 型単位構造，後者を T 型単位構造と呼ぶ．図 2(a) は，これらを図示したものである．

そして，大きさが入力構造 $\mathcal{I}$ 全体と単位構造の間にあたる依存構造集合という単位構造の集合を定義する．各依存構造集合は，折りたたみ最中で影響しあう可能性がある単位構造の集合である．図 2(b) は，二つの H 型単位構造を持つ依存構造集合を表している．図 1(c) は二つの依存構造集合を表し，水色の依存構造集合は二つの H 型単位構造から成り，緑色の依存構造集合は一つの T 型単位構造から成る．



(a) 入力形状　(b) 構造　(c) 二つの依存構造集合　(d) パッチに対する操作

図 1. 既存手法 [2] の手順



(a) 単位構造　(b) 依存構造集合の例

図 2. 単位構造と依存構造集合

折りたたみ方法

折りたたみ方法は，二つある．一つは，折りたたむ対象である可変パッチとそれと連結な固定パッチの連結部分にヒンジを挿入し，可変パッチを倒す方法である．もう一つは，可変パッチとそれと連結な固定パッチの連結部分のヒンジに加え，可変パッチ内部にヒンジを挿入し，固定パッチに対して垂直に折りたたむ方法である．

最適な折りたたみ可能な解

構造 $\mathcal{S}$ に対する解 $F_{\mathcal{S}}$ は，$\mathcal{S}$ 内のパッチがどのように修正，変形されたかで構成される．パッチ p に対する操作 m_p に基づき，どのような修正や変形が良いかを決めるコスト関数を以下のように書くことにする．

$$\mathrm{cost}(F_{\mathcal{S}}) = \sum_{p \in P_{\mathcal{S}}} \lambda_p \mathrm{cost}(m_p)$$

λ_p は，パッチ p の重要性を表し，入力構造 $\mathcal{I}$ に対する折りたたみ可能な最適解を以下のように定義する．

$$F^*_{\mathcal{I}} = \underset{F_{\mathcal{I}} \in \mathcal{F}_{\mathcal{I}}}{\mathrm{argmin}}\, \mathrm{cost}(F_{\mathcal{I}})$$

$\mathcal{F}_{\mathcal{I}}$ は，構造 $\mathcal{I}$ のすべての折りたたみ実行解の集合であり，$F^*_{\mathcal{I}}$ は，この問題の最適解を表す．

ここで，折りたたみ解の良し悪しを決定するコスト関数 $\mathrm{cost}(F_{\mathcal{S}})$ について述べる．構造 $\mathcal{S}$ は，単位構造の集合であり，$\mathcal{S}$ が持つ単位構造すべてに対してコストを決定する．単位構造は，必ず一つの可変パッチ p を持っているため，それに対する操作 $m_p = (N_h, s, a_0, a_1)$ で解を定義する．N_h はヒンジ挿入数，s はそのパッチが左右どちらに傾斜するか，a_0, a_1 は，縮小した際の端点を示す．例えば，$(a_0, a_1) = (0, 1)$ のときは，元のパッチの大きさを維持することを表し，反対に，$(a_0, a_1) = (0, 0)$ のときは，対象のパッチを完全に消去することを意味する．したがって，単位構造 B の解 F_B に対するコスト $\mathrm{cost}(F_B)$ は以下のように定義する．

$$\mathrm{cost}(F_B) = \mathrm{cost}(m_p) = \alpha N_h / \hat{N}_h + (1 - \alpha) {R_s}^2$$

以上から，ヒンジ挿入数が多ければコストは大きくなり，縮小率が大きければコストが大きくなる．このコストが最小な解を求めることが目的である．

2.2 アルゴリズム

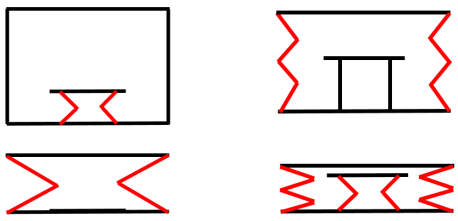
ここでは，分割された各依存構造集合ごとでの最適解の求め方，それらの解の組み合わせである最終的な解である入力物体全体としての最適解を求めるための組み合わせの方法について示す．

依存構造集合 U は，単位構造の集合 $\{B_i \mid i = 1, \cdots, N\}$ であり，U の解 F_U は $\{F^*_{B_i}\}$ の一つの組み合わせである．$[{t_i}^0, {t_i}^1]$ を B_i に対する時間間隔とすると，F_U は以下のようになる．

$$F_U = \{(F^*_{B_i}, {t_i}^0, {t_i}^1)\}$$

U は，各単位構造 B_i に対する操作の実行解の集合 $\mathcal{F}_{B_i}$ を保持する．U の解 F_U は，単位構造ごとに最適解を求め，各単位構造の最適解の組み合わせである．単位構造の解の組み合わせを求める際，折りたたみ可能な形状を求めるために単位構造の解どうしが互いに干渉し合わない解を求めるための衝突グラフを以下に定義する．

単位構造集合 $\{B_i\}$ を持つ依存構造集合 U が与えられたとき，その衝突グラフを，無向グラフ $G^{\mathrm{conf}} = (V, E)$ と表す．ここで，$V = \{v_j\} = \bigcup_i \mathcal{F}_{B_i}$, $E = \{e_{j,k} | v_j \text{ and } v_k \text{ conflict}\}$ とする．頂点集合 V は，U が持つ単位構造のすべての実行解であり，頂点間に辺が置かれるときは，二つの解が衝突するときである．このグラフを入力として，重み付き最大独立集合問題を解くことで依存構造集合 U の解 F_U を得る．



(a) 内部からの折りたたみ　(b) 外部からの折りたたみ

図 3. 折りたたみ順序

入れ子型最適化

入力構造 $\mathcal{I}$ は依存構造集合の集合 $\{U_i \mid i = 1, \cdots, M\}$ に分割され，$\mathcal{I}$ の解 $F_{\mathcal{I}}$ は，$\{F^*_{U_i}\}$ の組み合わせとして表すことができる．二つのループによる最適化により $F_{\mathcal{I}}$ を求める．

内部ループでは，依存構造集合どうしの衝突しない依存構造集合の組合せを求める．ある二つの依存構造集合の解どうしが衝突した場合は，その解を削除する．衝突グラフ上でその頂点を削除し，重み付き最大独立集合問題を解き直す．

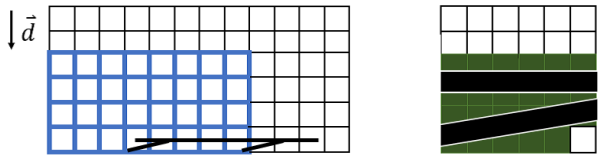
外部ループでは，最小の操作量を得るための依存構造集合の折りたたみ順序を決定する．依存構造集合の解のコストを考慮した一つ先を見た貪欲法を用いる．時刻 t において，U_p を折りたたむとし，その依存構造集合の解のコストを $C_1(U_p)$ とする．また，折りたたまれていない各依存構造集合の集合を，一つずつ時刻 $t+1$ で折りたたむときの解のコストの合計を $C_2(U_p)$ とし，$C_1(U_p) + C_2(U_p)$ が最小の折りたたみ順序を選択する．図 3 では，二つの依存構造集合の折りたたみ順序による操作量の変化を示している．(a) では，内部の依存構造集合から折りたたむが，二つの依存構造集合それぞれに一つのヒンジが挿入されている．一方，(b) の折りたたみ順序では，外部に三つのヒンジが挿入されており，(a) に比べて操作量が増えていることがわかる．

3 提案手法

既存手法 [2] を基にした折りたたんだ後に物体が占める領域を考慮した折りたたみ手法と，効率良く 3D プリンタで物体を造形する手法を提案する．

3.1 折りたたんだ後を考慮した折りたたみ手法

物体を折りたたむときに，どのような形状に収めるかを指定することで，空間をより効率良く利用できる．提案手法では，入力に，既存の入力である物体と折りたたみ方向に加え，直方体の領域を追加する．この領域内に最終的に折りたたまれた状態の物体が収まることを目標とする．ボクセル化手法を利用して，折りたたみ後の領域を考慮した折りたたみ可能な形状を求める手法を提案する．



(a) 入力領域とボクセル　(b) サブボクセル

図 4. ボクセル化の適応

3.1.1 ボクセル化によるコスト

各依存構造集合を折りたたむとき，重み付き最大独立集合問題から得られた解が与えられた領域をどのくらい占めているかを離散的に評価する．図 4(a) のように，折りたたみ方向 $\vec{d}$ に対して入力形状を折りたたむときに占める領域を十分に確保してボクセルを作る．図 4(a) には，入力領域である青いボクセルと入力構造を図示する．(b) は各ボクセルが持つ $6 \times 6 \times 6$ 個の立方体を表す．この小さいボクセルをサブボクセルと呼ぶ．(b) の濃い緑色のサブボクセルは物体と交差するサブボクセルである．依存構造集合 U に対して，以下のようにコスト $C_V(U)$ を定義する．

$$C_V(U) = \sum_{v:v\in V, v\cap U\neq\emptyset} \left(1 - \frac{\text{volume}(v\cap U)}{\text{volume}(v)}\right)$$

v は一つのサブボクセルを意味する．また，volume(v) は，v が持つサブボクセルの数．つまり，volume(v) $= 6 \times 6 \times 6$ であり，volume($v \cap U$) は，v 内のサブボクセルと物体 U が交差しているサブボクセルの数とする．ここでは，局所的に見てその依存構造集合が領域内に収まるのならば良い折りたたみとすることから，領域内外でコストに異なる重みを加える．領域内のボクセルに対してのコスト $C_{V\text{in}}$ は C_V を用いて，$C_{V\text{in}} = 1/3 \times C_V(U)$ とし，領域外のボクセルに対してのコスト $C_{V\text{out}}$ は，$C_{V\text{in}} = 1.0 \times C_V(U)$ とする．

このコスト関数を用いて，指定した領域になるべく収まるような形状を生成する．既存手法 [2] の入れ子型最適化での依存構造集合の解を決定する際に，このコストを追加することで，折りたたんだ後に占める物体の領域を考慮した最小コストを持つ解を決定する．

3.2 3D プリンタでの造形手法

本研究では，折りたたみ可能な形状を得た後，その形状を実際に 3D プリンタで印刷し，その 3D プリンタでの効率の良い造形手法を提案する．既存研究では，折りたたみ可能な形状を得た後，その形状を入力として 3D プリンタで印刷している．既存研究や本研究では FDM 方式 3D プリンタを用いており，これは，一層一層積み重ねて造形していく方法で，入力物体の高さが造形時間に大きく影響する．

また，この方式では造形の際に，水に浸すと溶けるサポート材という出力物体の不安定な部分を補助する材料を用いることができる．一つは物体を造形する材料を押し出し，もう一つはサポート材を押し出す二つのヘッドによりサポート材を利用し，一層一層積み重ねて物体を造形する．本研究では，この造形時間の短縮とサポート材の使用量の削減を目的とした造形手法を提案する．

3.2.1 多角形の平面詰め込み問題

既存手法を基にして得られた折りたたみ可能な形状を各部品を個別に造形しても組み立てられるように分解する．まず，一つの依存構造集合を一つの部品として考える．加えられた操作からヒンジが挿入されているパッチに対しては，ヒンジが挿入されている部分で部品を分け，二つの部品とする．このように一つの形状を部品に分解する．

次に，各部品を多角形に近似する．入力は家具のような板状の物体を仮定しているため，立方体のように，どの方向から見ても高さがある物体は想定していない．そのため，どの部品も上手に回転させれば，高さを考慮しなくてよいと見ることができるため，部品を一つの平面上に並べる．その際，各部品に対して凸包をとり，部品を多角形として近似する．その多角形を

入力として，多角形の平面詰め込み問題を解き，効率の良い部品の配置を決定する．3D プリンタの特性から，部品はなるべく密集していると造形時間を短縮できることから詰め込み問題を利用する．

3.2.2 bottom-left 法

本研究では，多角形の詰め込み問題を [1] で述べられている bottom-left 法で解く．ただし，ここでは各多角形の回転は許さないものとする．入力として，幅 W，高さ H の大きさを持つ造形可能領域 R と，n 個からなる多角形（部品）集合 $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ が与えられるとする．多角形 $P_i \in \mathcal{P}$ は，母材 R 上に配置され，多角形対 $P_i, P_j \in \mathcal{P}$ は互いに重ならないという制約条件を満たす各多角形 $P_i \in \mathcal{P}$ の配置を決定する．この二つの制約条件の判定方法として，NFP(No-Fit–Polygon) を用いる [1]．多角形 P, Q に対して，それぞれある点を参照点として，多角形を参照点からの相対位置で表す．P の位置を固定したとき，P と Q が重なるような Q の参照点の位置すべてを Q の P に対する No-fit-Polygon と呼び，NFP(P, Q) と表す．

NFP はミンコフスキー差と等価であると知られており，本研究では計算幾何ライブラリの CGAL を用いて，ミンコフスキー差から NFP を計算した．

bottom-left 法では，なるべく左下に一つずつ多角形を詰め込む．高さが低い候補点を優先し，高さが同じのときは，幅の値が小さい点を優先する．NFP$(\bar{R}, P)$ から，すでに配置されている多角形と，詰め込もうとする多角形 P の NFP の和を差し引いた領域の境界の頂点が候補点とする．この候補点から，重ならない配置を決定する．bottom-left 法は，詰め込む順番に解の良さが依存するため，本研究では，ランダムな順序でいくつかの配置を決定し，最も造形物の高さが低い配置を結果として利用する．

4 実験結果

以上の手法を，Intel(R) Core(TM) 3.40GHz，RAM 16.0GB という実験環境の下，実装した．ここで，各パラメータは，折りたたみ方向 $\vec{d} = (1, 0, 0)$，各パッチに対して最大ヒンジ挿入数 $\hat{N}_h = 3$，縮小率 $N_S = 4$，α は，0.5 とした．また，三次元形状を実際に出力するための 3D プリンタは，FDM 方式の Ultimaker3 Extended を利用し，造形可能領域は，197×215×300mm である．

図 5 に入力形状と折りたたみ可能な形状，部品の配置を示す．図 5(c) に示した部品の配置を求めるまでの計算時間は，119 秒 46 であった．この結果の 3D プリンタでの造形には，10 時間 34 分かかり，図 5(b) のまま 3D プリンタで造形する場合は，34 時間 19 分かかるため，部品の配置によって造形時間の短縮ができた．サポート材消費量は，部品に分割しない場合は 50g であったが，部品に分割し，bottom-left 法から配置を決定したものでは 7 g となった．したがって，サポート材の削減ができたと言える．

また，実際に 3D プリンタで椅子を入力として造形したものを図 6 に示す．この図での結果は，造形時間は 16 時間 47 分であり，サポート材消費量は 7g であった．

(a) 入力形状　(b) 折りたたみ可能な形状　(c) 部品の配置

図 5. 提案手法での出力結果

(a) 分解した部品　(b) 組み立てた物体　(c) 折りたたんだ物体

図 6. 3D プリンタでの造形

5 まとめと今後の課題

3D プリンタでの造形に対して，配置を決定することで，サポート材の消費量の削減と造形時間の短縮ができた．また，指定した領域内になるべく収まるような形状を求められ，より空間を効率良く利用するための折りたたみ物体をデザインできると考えられる．

今後の課題として，部品の配置の改善が挙げられる．自由に回転を許す多角形の詰め込み問題の解法を用いることで，より効率の良い配置が可能になる．また，3D プリンタでの出力物体が構造上不安定なことがあるため，物体の厚みやヒンジの挿入場所を考慮し利用できる折りたたみ可能な物体を出力するための指標を作成することも重要であると考える．

謝辞

本研究を進めるにあたり，適切な御指導，御指摘をして頂きました中央大学理工学部情報工学科の今井桂子教授，明治大学先端数理科学インスティテュートの森口昌樹特任講師に心から感謝致します．また，今井研究室および情報工学専攻の学生各位の協力なしには，この論文は完成しませんでした．深く感謝致します．

参考文献

[1] K. A. Dowsland, S. Valid and W. B. Dowsland, “An alogrithm for polygon placement using a bottom-left strategy,” *European Journal of Operational Research*, 141:371–381, 2002.

[2] H. Li, R. Hu, I. Alhashim, and H. Zhang, “Foldabilizing Furniture,” In *Proceedings of ACM SIGGRAPH 2015*, 34(4):90:1–12, 2015.

[3] X. Li, T. Ju, Y. Gu, and S. Hu, “A Geometric Study of V-style Pop-ups: Theories and Algorithms, ” In *Proceedings of ACM SIGGRAPH 2011*, 30(4):98:1–10, 2011.

[4] Y. Zhou, S. Sueda, W. Mattusik, and A. Shamir, “Boxelization: Folding 3D Objects into Boxes,” In *Proceedings of ACM SIGGRAPH 2014*, 33(4):71:1–8, 2014.