

積上げ式凹凸マップの構築と制御と最適化

今給黎 隆 (正会員)

原 寛徳 (正会員)

東京工芸大学 芸術学部

Construction, Control and Optimization of Stacked Relief Maps

Takashi Imagire (Member)

Hironori Hara (Member)

Faculty of arts, Tokyo Polytechnic University

{t.imagire, hara} @ game.t-kougei.ac.jp

アブストラクト

NICOGRAPH 2020 で展示を行った「積上げ式凹凸マップによるディザリングの立体化」の実装の解説及びその拡張についての研究である。NICOGRAPH 2020 では一辺が 18cm の正方形上の平面を 64×64 のブロックに分割して、単色プリンターで印刷したオブジェクトを積み重ねて構築する凹凸が付いた絵を展示した。しかしながら、解像度を高くしたり、絵の様相が複雑になると、積み上げる形状を計算するための時間やメモリが多くなり、幅優先探索や深さ優先探索では解の発見が困難になる。今回は、形状を計算するための探索方法について調査を行い、エニータイムビームサーチ法によって、近似解ではあるが十分に正確で、高速に解を得ることに成功した。また、実際上は、積む順序を制御したいという要望が生ずる。今回は、探索時のグラフ構造を有向グラフにすることで積み上げる順を制御する手法を提案する。さらに、印刷物の厚みを削減する方法を提案する。

Abstract

This is a description of the implementation of the "Three-dimensionalization of Dithering on Stacked Concave-Convex Map" that was exhibited at NICOGRAPH 2020 and a study of its extension. At NICOGRAPH 2020, we exhibited an uneven picture constructed by stacking objects printed on a single-color printer by dividing a plane of 18 cm square into 64×64 blocks. However, as the resolution increases or the pattern of the picture becomes more complex, the time and memory required to compute the stacked shapes increases, making it difficult to find a solution using width-first or depth-first search. In this study, we investigated search methods for computing the shape, and succeeded in obtaining an approximate but sufficiently accurate and fast solution using the anytime beam search method. In addition, there is a practical need to control the order of stacking. We propose a method to control the stacking order by using a directed graph structure during search. Furthermore, we propose a method to reduce the thickness of printed materials.



図1. NICOGRAPH 2020 での作品展示。上段中央が元絵。上段右が提案手法のRGB各1bit量子化の平均ディザリング。下段は左から同モノクロの誤差拡散ディザリング、パターンディザリング(Bayer)、1bit量子化の平均ディザリング。

1. はじめに

NICOGRAPH 2020 において、凹凸のある絵に関する作品を発表した(図1) [1]. この作品は単色の3Dプリンターで印刷したオブジェクトを重ねて作成された立体的なカラーの絵である. 本作品は, ディザ法を理解するための可視化という教育目的で制作を開始した. 図2のような部品を下から上に積み上げることで図3のような作品が出来上がる. この作品を制作する上での挑戦は, 重ねるオブジェクトの個数を減らすことである. 誤差拡散法やBayer行列でのディザ法の可視化では細かなドットが表れやすい. 一つ一つのドットを個別の部品として, 穴をあけておいた場所に差し挿む方法も考えられるが, 部品の個数が増えやすく, 管理が煩雑となる. 個々のドットを別の部品の後ろで繋ぐことでオブジェクトをまとめられ, 部品の数を減らすことができる. これによって部品の管理がしやすくなり, 組み立てが簡単になる. ここで, どのようにすれば部品の数を減らせるのかということが問題となる. この問題は, 問題の設定自体が特定の作品を対象としたものであるため自明ではない.

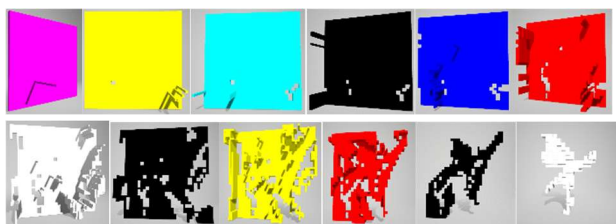


図2. 解像度40×40のLennaの1bit量子化の平均ディザリング. 左上から右下に順に積み上げることで図3となる.

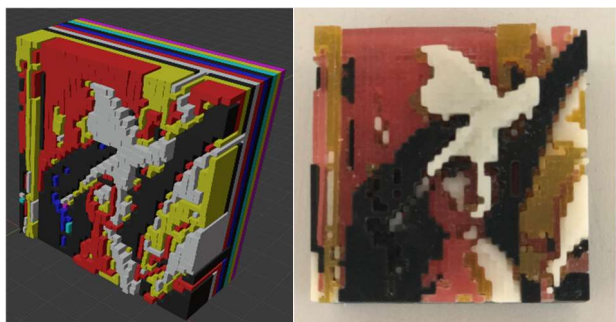


図3. 解像度40×40のLennaの8色での平均ディザリングの積上げ式凹凸マップのモデル(左)と実物写真(右).

NICOGRAPH 2020の作品では, 絵画のディザ法の可視化として作品を制作したが, 本手法は凹凸のある2次元データの立体化の一つの手法として有用である. 本手法の応用に地形模型が考えられる. 測量されたそれぞれの場所の高さに応じて地図に凹凸をつけ, 土地の起伏を把握しやすくする地図である. 地形模型は古くから作られているが, カラーの地図を凹凸のある形状として印刷するには高価な3Dプリンターが必要である. 本手法を用いることで, プラモデルのように固定色の部品を印刷して, 上に載せるだけの簡単な組み立てでの地形模型が実現できる. また, この作品の発展として透過に注目をした. 火山があれば赤く光らせたいし, 夜の街の表現であれば電灯を光らせたくな

らであろう. このような場合は, 光らせる部品を透明な部材で印刷して, 下から光源で照らせば良い. この場合, 透明なオブジェクトの下に不透明なオブジェクトがあると光が届かなくなるため, 透明なオブジェクトは一番下に配置する必要がある. このような部品の上下の指定は, NICOGRAPH 2020の作品よりも複雑な計算が必要となる.

2. 関連研究

本研究の目的は, 高さ情報を持つ色付き2次元データを単色の3Dプリンターで出力する方法の実現である. 作品[1]はNICOGRAPH 2020に出展するために作成したものであるが, 特別なものではない. 凹凸のある図は, 触地図として視覚障碍者が地理を把握するために活用されている. Taylorら[2]は, スマートフォンと組み合わせたインタラクティブな触地図を提案している. 穴の開いた3Dプリンターの印刷物を用いることで, その下に置かれたタッチパネルによる反応を実現でき, 下からの映像も組み合わせることができる. しかしながら, 一つの3D印刷物を用いるため, 図の上下や左右を横断するような穴を空けることができない. Auricchioら[3]は, 美術館の案内図を3Dプリンターによる複数の色の部品を組み合わせて実現した. その制作過程は明示的に説明されていないが, 印刷後の組み立てに6~7時間かかっており, 組み立ては容易ではない. Urbasら[4]は, 単色の3Dプリンターにおいてフィラメントを交換しながら上に重ね書きすることで, カラーの触地図を制作した. Urbasらの手法は, 一つの印刷物として処理をする形式であり, 複数の3Dプリンターを用いた並列処理での高速化はできない.

触地図を作るためのツールとして, BrownとHurstによるVizTouchが提案されている[5]. VizTouchは2次元関数を触覚的に理解するための3Dモデルを自動で作ることができる. ただし, 多層のモデルを構築することはできない. また, 類似のテーマとして, 立体的なパズルを自動で計算する手法が提案されてきた[6]-[8]. 今回は, 組み立てやすさの理由から, 上に重ねれば組み立てられるという簡単な構造を対象とする. この場合, 既存の手法をそのまま適用することはできない.

3. 提案手法

3.1 問題の定式化

今回の対象と問題を定式化し, その解法を検討する. 複数に分割した部品を, 特定の順番で上に積み上げることで一枚の凹凸のある絵を構築する. それぞれの部品は色ごとに3Dプリンターで印刷する. したがって, 対象の絵を色ごとに分割する. ただし, 分割した部品の数はなるべく減らしたい. 例えば, ある色の部品の上に2つの部品を載せている場合を考える(図4中央). この際, 土台となる部品の上に別の部品を載せれば(図4左), 3点の部品へと分割されるが, 2つの部品を裏から通して繋げば(図4右), 部品を2つに減らすことができる. 今回は, それぞれの部品を下から積み重ねる形式の組み立て方法を対象とし, その組み立て方で部品数を最小とする分解の仕方を導く間

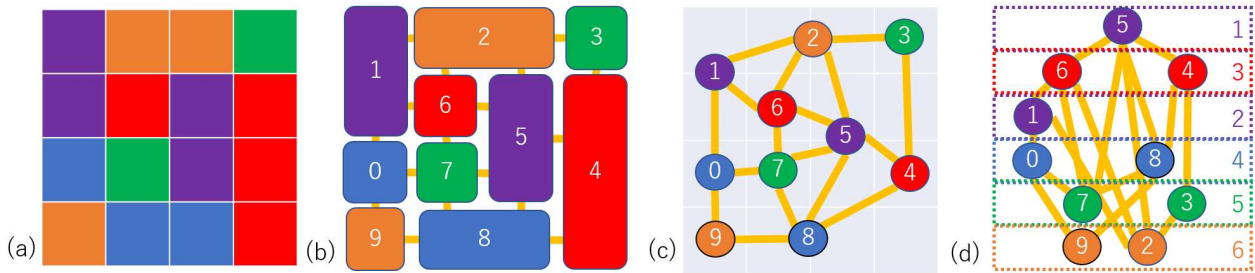


図5. 画像のグラフ化と並び替え. (a)元の画像. (b)グループ化と接続情報の構築. (c)グラフ化. (d)レイヤー分け

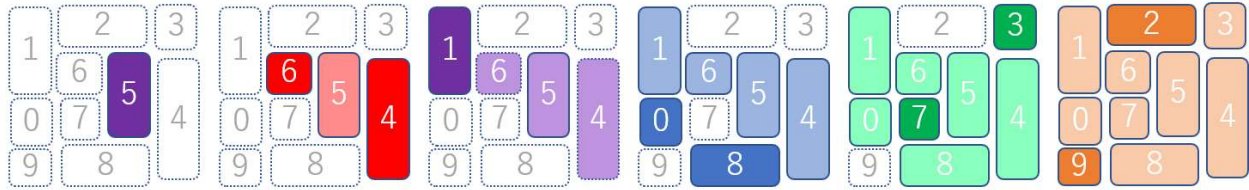


図7. 各レイヤーで印刷するグループ. 右の出力ほど下に置かれる. 薄い色は下に隠れる底面で濃い色が上から見える表面.

題とする. なお, 小さな部品を上には置くのは, 大きな部品を上には置くよりも細かな作業であり神経をすり減らす. また, 小さな部品が上に置かれると, 固定されにくいため倒れたり, 紛失しやすくなる. このため, 小さな部分ほど下で繋がるような3Dモデルを構築する.

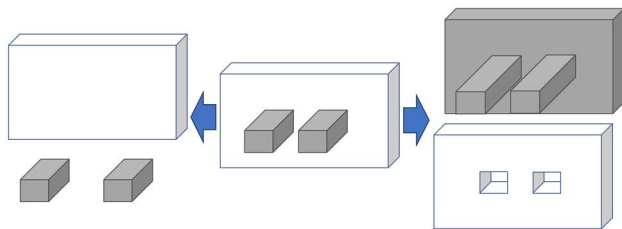


図4. 部品の分割例. 中央の絵は左のように分けると3つに分割され, 右のように分けると2つに分割される.

3.2 制約のない形式の解法

本問題を解くために画像からグラフ構造を構築する. 画像について上下左右で繋がっている同じ色の画素を一塊のグループとする. このグループをノードとするグラフを構築する. ノード間のエッジは, グループ内の画素と隣接する画素を持つグループの間に設ける. 図5がグラフの一例である. 図5(a)を変換元の画像とする. 隣接している同じ色の画素を一つのグループとしたものが図5(b)である. ここでは, 各グループにIDとなる数値を記した. 黄色い線は, グループ間に隣接する画素があることを示している. グループをノードとして無向グラフとして明示的に記述したのが図5(c)である.

構築したグラフ構造から, 凹凸マップの部品を構築する方法を考える. グラフ構造に上下の軸を導入し, 作成した部品の位置の上下をグラフの上下と考える. この上下構造をレイヤーと定義する. 図5(d)がノードのレイヤー分けの結果である. 図5(d)の右の数字は, 上から数えたレイヤーの段数である. 3Dプリンターで印刷するのは単色の印刷物であり, 横方向に同じ色のノードが存在できる. ただし, これらのノードは一つの印刷物と

して印刷できなければならない. これを実現するには, 同じレイヤーにあるノードは, 自分よりも上のレイヤーのノードの下を回り込めばよい. これは, 各レイヤーの全ノードが自身のレイヤーよりも上のレイヤーのノードを伝えて繋がる事と等しい.

レイヤー構造を構築した際に, レイヤーの数が少なければ, 3Dプリンターで印刷した際の部品数が少なくなる. レイヤー数を最小化するために最適化関数を導入する. ただし, 単にレイヤー数を最小化するのではなく, 組み立てやすさを考慮して, 大きな部品が上に配置される最適化関数を導入する. 具体的には, 次の(1)式での目的関数 E を最小化する解を検索する.

$$I = \text{minimize } E, \quad E = \sum_{x,y} d(x,y). \quad (1)$$

ここで, $d(x,y)$ は画素 (x,y) が属する部品を重ねる際のレイヤーの段数である. 広い面積が下に行くほど d の和は大きくなるため, d の和が小さいほど, 小さな画素の塊は後ろに配置される. 具体的な d の値を図6に示した. 図5(d)で決めた各ノードのレイヤーの段数をノードに対応する画素の位置に記述した. この場合の目標関数 E の値は, 各画素におけるレイヤーの段数の合計なので, 図6に書かれた数字の総和である56となる.

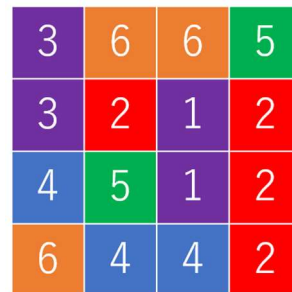


図6. 各画素のレイヤー段数. 合計値が $E (= 56)$.

図7は各レイヤーで印刷するグループの可視化である. 濃い色のグループは各画素の高さで印刷し, 薄い色のグループは, ノードを裏で繋ぐように一段上のグループの下の高さで底面として印刷される. 底面は印刷する際に自重で折れたり曲がらない

厚さとした (NICOGRAPH 2020 の作品では2mm) . レイヤー数が増えるほど、この底面の厚さの総和は増加する。その物理的な厚みを減らすための改善を3.4節で提案する。

レイヤー分けの中心的な計算は、探索である。以下がレイヤー分けの探索アルゴリズムである。アルゴリズムでは、探索の候補となる「オープンリスト」、既に処理を行ったノードを格納する「クローズドリスト」と、レイヤーを下方向に保持する「レイヤースタック」を導入する。

1. ノードを一つ取り出す (最上位ノード) .
2. 最上位ノードをオープンリストに追加する。
3. オープンリストに含まれるノードの色を一つ選ぶ。
4. 選んだ色のノードをオープンリストの中から抽出する。抽出されたノードはオープンリストから削除する。
5. レイヤースタックにレイヤーを追加する。選んだ色をレイヤーの色とし、抽出されたノードを関連付ける。
6. 今まで抽出されたノードとつながっているノードで、まだオープンリストにもクローズドリストにも登録されていないノードをオープンリストに追加する。
7. 抽出されたノードをクローズドリストに追加する。
8. オープンリストにノードが残されていれば3に戻って繰り返す。
9. 目的関数 E の値を計算して、結果が最小値であれば、今回の構造 (最上位ノードとレイヤーの色の順番) を最適解の候補とする。
10. 1に戻って最上位ノードや各レイヤーで選択する色を変更しながら最適解を探索する。

このアルゴリズムの説明は完全ではない。最上位ノードやレイヤーの色を選択する方法を定めていない。これは探索アルゴリズムを選ぶ自由度を与える。最上位ノードや色の選択について、幅優先探索(BFS)として、最上位ノードを順次選択するとともに、レイヤーの色を上から順に枝分かれしながら探索したり、深さ優先探索(DFS)として末端から色を変えながら探索する方法が考えられる。幅優先探索は最適解の候補を発見すれば、その深さの全ての組み合わせを探索することで最適解が求まるが、使用メモリが大きい。深さ優先探索は使用メモリが少ないものの、全てを探索しないと最適解が判明しない (その時点の最適解の候補のレイヤーの深さで探索を打ち切る枝刈りが可能であるが、最も少ないレイヤー数がいつ見つかるのかは分からない) .

深さ優先探索は、レイヤーの数が少ない最適解の候補を早く見つけるほど探索時間を短くすることができるが、このような良い最適解の候補の早い探索にモンテカルロ木探索の利用が考えられる。モンテカルロ木探索の根ノードとして、各ノードを最上位ノードとする探索ノードを登録する。各探索ノードの状態からオープンリストにノードがなくなるまでランダムに色を選択し続ける (プレイアウト) . プレイアウトを繰り返し、UCB値[10]が下記(2)式の閾値 V_{UCB} を超えた際に、探索ノードをそのノードから繋がるレイヤーの色で探索ノードを成長させる。

$$V_{UCB} = w_j + \sqrt{\frac{2 \log n}{n_j}}. \quad (2)$$

ここで n は全プレイアウト数、 n_j は現在の探索ノード j のプレイアウト回数である。今回は、下記の最適解の候補レイヤーの数 D_c とプレイアウト中のレイヤーの数の最小値 D_j の差を正規化した値をUCB値の勝率 w_j に用いた。

$$w_j = 1 - \frac{D_j - D_c}{D_j} = \frac{D_c}{D_j}. \quad (3)$$

勝率 w_j は、最適解と同じレイヤー数になる時に1、そこから値が大きくなるほど0に近づくなるべくシンプルな関数として採用した。他の関数の形式も想定しうが、本関数の式で問題が生じていないため、本研究では(3)式の関数を採用している。勝率 w_j に目的関数 E も考慮した方が良いと思われるが、今回は、後の説明によるDFSの組み合わせにより目的関数 E の効果を取り込む。なお、モンテカルロ木探索は確率的な探索であり、探索した最適解候補が最適解であるかどうかは不明である。今回は、10万回のプレイアウトで新たな最適解候補が見つからなかった探索ノードを削除することで計算を打ち切り、探索を終わらせることにした。モンテカルロ木探索で得られる解は近似解であるが、10万回のプレイアウト回数の上限でも比較的早い時間で最適解が得られた。なお、10万回という最大回数はad hocな数値であり、計算時間が長くなりすぎない値として設定した。DFSとモンテカルロ木探索は末端で組み合わせしており、現在の探索ノードのレイヤー数が最適解の候補のレイヤー数よりも4階層浅い場合には、DFSで残りの下位のレイヤーの全検索を行い、不確実性を減らしている。

それ以外の探索方法を模索したところ、エニータイムビームサーチ法[11]が適用できた。ビームサーチ法は、幅優先探索の各深さにおいて、決められた数 (ビーム幅) の成績が良いノードだけを残して探索を深める方法である。幅優先探索と異なり、メモリ使用量を制限しながらの探索が行える。今回は、それぞれの深さで色を選択した後の残りノード数をレイヤーの色の選択の指標として用い、残りノード数が少ない色を成績の良いレイヤーとして残した。ビームサーチ法は近似解を得るための手法であるが、ビーム幅が広いほど良い解が得られやすい。エニータイムビームサーチ法はビーム幅を広げながら再探索を行う事で、より良い解を探索する。エニータイムビームサーチ法は、ビーム幅を無限に広げることができるとすると、幅優先探索と一致する。逆に言えば、ビーム幅を広げるとメモリ使用量が増すため、ビーム幅を一定の範囲以上に広げるのは難しい。今回は、ビーム幅を設定した後、そのビーム幅で探索を行い、そのビーム幅を倍に広げて探索を続ける。探索を打ち切る条件は、3回ビーム幅を広げたとしても新たな最適化の候補が見つからなかった場合に探索をあきらめる形とした。

3.3 制約つき問題の定式化

透明な部品を使う場合を考える。透明な部品の下に不透明な部品が来ては困るので、一番下の部品を透明な色としたい。複数の透過色を用いる場合は、複数の透過色のレイヤーの上下関係を指定することが考えられる。この状況に対応するためにグラフを無向グラフから有向グラフに拡張する。下のレイヤーにしたい色を持つノードに繋がるエッジに対して、下のレイヤー

色のノードへ進む向きに有向辺の向きを設定する。レイヤー構築時に有向辺の向きが下のレイヤーから上のレイヤーに向かう矢印が現れたら、そのレイヤーの色の組み合わせは無効な状況として除外する。

この制約付きの探索では、未探索ノードが残っているにも関わらず探索するノードがない（無向もしくは有向グラフの順方向に繋がるノードがない）という状況が発生する(図8)。これは、(透明な)レイヤーのノードによって他のノードの繋がりが分断され、制約がない場合には下を通して繋がれていたものが、下を通ることができないために、オープンリストに登録できないノードが生じるためである。探索するノードが無いにも関わらず、未探索ノードが残っている場合は、未探索ノードの中からノードの一つを選択し、新たな最上位ノードとして探索をやり直す(図8の右側の最上位ノード)。これにより探索を最後まで続けることができる。異なる最上位ノードから始まるレイヤーの塊は、互いに接することがないので、独立して透明なレイヤーの上に位置することになる。したがって、ポリゴンモデルの作成時も、最初の最上位ノードから始まるレイヤー群(図8の左側)の高さから別の最上位ノードから始まるレイヤー群(図8の右側)の高さを引くことで底面の厚さを減らすことができる。

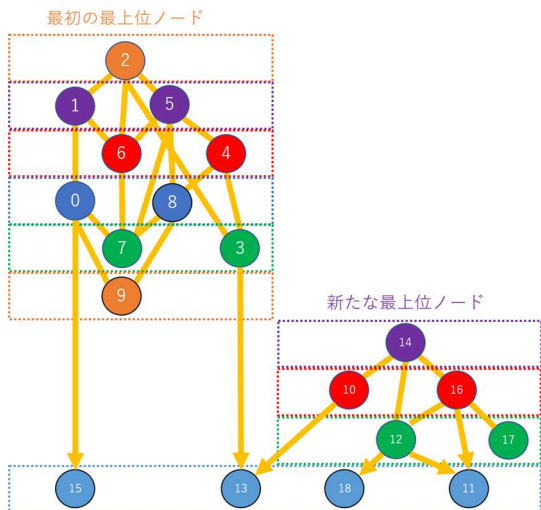


図8. 制約のある場合のレイヤー化の例.

3.4 物理レイヤー数の削減

3.2節で紹介した手法は、各部品に共通の底面を追加して、底面を使ってノードを裏で繋げる。そのため、生成されるオブジェクトの高さは「物体の最大の高さ+底面の厚さ×レイヤー数」になる。レイヤー数が増えると本来の凹凸の高さよりも底面の

厚さが目立つようになる。これは、地層のようなパターンが目につき格好が悪い。この節では、隣接する複数のレイヤーを一つにまとめることで底面を薄くする手法を提案する。

隣接するレイヤーを統合した場合、形状はそれぞれのレイヤーで高さを持つ柱の部分と底面の画素で構成される。図9(a)は、図7の右側2つのレイヤーを重ねた状態である(図9(a)では、濃い色の高さを持つ部分以外は全て薄い色の底面となっているが、一般には注目しているレイヤーの下のレイヤー部分の底面に穴が開く)。ここでは図9(a)の薄い色を複数のレイヤーで共有する事でレイヤーを統合する。レイヤーを統合するための制約として、画素の大きさよりも細かな構造にしないこととする。薄い色の部分を細かく分ければプリント基板の配線の様な複雑な形状で凹凸の柱部分を繋げることができるが、3Dプリンターで印刷する際の強度が心配になる。元画像の画素は3Dプリンターで形がくっきりと作れる大きさであり、丈夫なはずなので、画素の大きさを最小単位として底面を再構築する。なお、底面の強度を強めるために、底面の画素をなるべく近い場所の柱のレイヤーの色とすることで、局所的に太い形状のモデルを構築する。

統合アルゴリズムを説明する。図9(b)は、灰色の部分に底面であるが、グループ化されている表現を一度画素単位に分割をした。この画素を統合において繋いでいくノードとして同じ色のノード間を経路探索する。ただし、高さを持つ柱のノード(濃いノード)は、グループを探索のノードとし、画素に分解しない。経路探索には、図9(b)における橙色の各ノードを出発点とするような双方向ダイクストラ法[14]を採用した。結果は図9(c)である。図9(c)では最短経路を薄い橙色で塗りつぶした。また、最も近い濃い色のノードへのマンハッタン距離を各画素に記録した。次に、別の色のノードを接続する。まだ経路となっていない画素を経路探索して緑色のノードを繋いだのが図9(d)である。なお、図9では、それぞれのレイヤーで2つのノードを繋いだが、一般的にはレイヤーに含まれるノードは複数あり、全ての同じ色の柱のノードを繋がなくてはならない。各ノードが繋がっているかどうかは、繋がっているノードをUnion-Find木[15]で分類することで高速に判断できる。

最短経路を求めた後は、経路を広げていくことで、まだ決まっていない灰色の画素の色(レイヤー)を定めていく。底面の画素について、上下左右の画素を調べ、隣接する画素がレイヤーのノードか経路の画素であったら、そのマンハッタン距離が近い色(レイヤー)をその画素の色とする。ここで、隣接する画素を見た場合に、他の色のマンハッタン距離の方が近い場合がある(図9(d)の上から2行目、左から3列目の画素)。この場合、

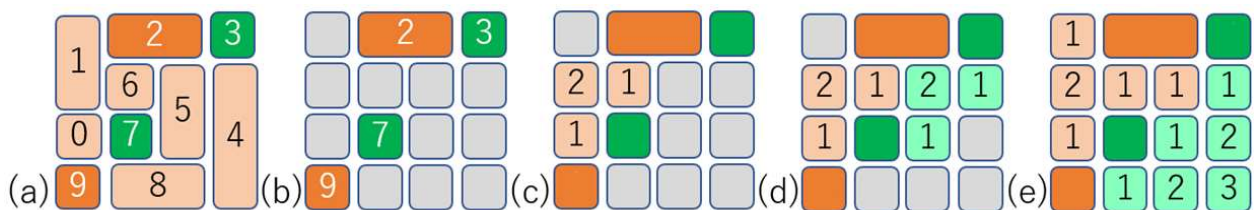


図9. レイヤーの重複の可能性の検索と画素の割り当て。(a)1番底と2番目の底の重ね合わせ。(b)上位レイヤーのノードの分解。(c)一番底のノード間の最短経路探索。(d)上のレイヤーの最短経路探索。(e)画素の割り当て。

その画素を別の色の画素とした場合に、その上下左右の同じ色の画素を別の経路で全て繋げられる場合に、より近いマンハッタン距離の色（レイヤー）に置き替える（図9(d)の2行3列目の成分は、緑色の2の数字から図9(e)で橙色の1に変わった）。これにより底面の幅が広がる。この拡張操作を繰り返して、未決定（灰色）の画素がなくなるまで処理をしたのが図9(e)である。1つの底面を、2つのレイヤーで共有することができた。

このように、隣接する複数のレイヤーについて、それらレイヤーの一番下の底面の画素を伝って同じレイヤーの全てのノードを単連結に繋ぐことができれば、それらレイヤーは一つの底面でまとめられる。一番下の2つのレイヤーから始めて、その2レイヤーが統合できれば、さらに上のレイヤーも一緒に統合できるのか探索していく。いずれかのレイヤーのどれかのノードが繋がらなければ、レイヤーは統合することができない。探索に失敗した場合に、最後に追加したレイヤーを取り除いて実際に統合する。次に、取り除いたレイヤーとその上のレイヤーについて統合できるのか新たに調べ直し、最終的に残りのレイヤーがなくなるまで検証する。なお、統合できるかどうかの判定の結果は、経路探索をするレイヤーの順番に左右される。したがって、経路探索をするレイヤー探索順を全列挙して全ての並び替えの順で検証を行う。実際の統合は、最初に成功した順番のレイヤーの探索順を採用した。全体的なレイヤーの統合のアルゴリズムの流れはアルゴリズム1のようになる。

`next_permutation` は、辞書順での次の順列を生成する関数である。

アルゴリズム 1 レイヤーの統合の全体的な流れ

```
function レイヤーの統合
  s ← layer数
  n ← 2
  while n < s
    array ← (s-n, s-n+1, ..., s)
    do
      for all l ← array do
        レイヤーlのノードを繋ぐ
        if いくつかのノードが繋がらなかった then
          go to repeat
        end if
      end for
    end for
    go to add_layer
  repeat:
    while next_permutation(array)
      s - n + 2 からsまでのレイヤーを統合
      s ← s - n + 1
      n ← 1
    add_layer:
      n ← n + 1
    end while
    s - n + 2 からsまでのレイヤーを統合
  end function
```

4. 結果

提案法により複数のモデルを構築した。モデルの構築に使用したPCは、Intel Core i7-8700K CPUに32GBのメモリのPCである。アプリケーションはx86版でビルドした。これは、利用した画像

読み込みライブラリの制約によるものである。今回の計算では、GPUやマルチスレッドでの並列化は行っていない。最も負荷が高いレイヤー化の処理は、最適解の候補の更新について排他制御が必要な点に注意すればマルチスレッド化は比較的容易であるが、CPUコア数に比例する効果しか得られないため、処理時間はアルゴリズムを並列化していないケースで計測をした。

4.1 Lenna

Lennaの画像をディザ法により減色した結果が図1の展示である。下の段は白と黒の2色による表現で、右から平均ディザリング法、Bayer行列による配列ディザリング法、誤差拡散法による結果であり、右上は8色での平均ディザリング法の結果である。64×64ドットの解像度で表現し、18×18cm²の大きさで印刷を行った。なお、3Dプリンターで印刷する場合は、フィラメントやレジンを膨らむため、上位のレイヤーと接触する面において0.3mmほど内側に縮めた形状のポリゴンモデルを出力した。

表1は、図1および画像の解像度を変化させて探索時間を調査した結果である。レイヤー化の探索時間だけを計測しており、画像の読み込み時間やポリゴンモデルの書き出し時間は含まれていない。時間が短いほど良い結果である。白と黒のモノラル画像に対する検索では、レイヤーの色は上位のレイヤーとは異なる色を選択するだけなので、全てのグループを最上位ノードとする探索を一度行えば良く、シンプルなDFSやBFSが高速な結果となった。色数が増えるとDFSでは計算時間が増大し、BFSでは2GB以上のメモリを確保しようとしてアプリケーションが強制終了した。モンテカルロ木探索やエニータムビームサーチ法では、エニータムビームサーチ法の方が10倍から100倍程度高速であり、共に現実的な時間で計算を終了できている。モンテカルロ木探索やエニータムビームサーチ法で得られる解は近似解である。今回の範囲では、ビームサーチ法では、64×64解像度の8色の場合に最適解と同一のレイヤー数が得られていたものの目的関数Eの値が最適解よりも大きな（悪い）値の最適化候補しか得ることができなかった。一方、モンテカルロ木探索では最適解が得られた。

画像の解像度を大きくして実験した際の追実験も行った。表2が、大きな解像度にした際の結果である。この追実験では、PCの調達の都合により、AMD Ryzen™ 7 7735HS CPUに16GBのメモリを持つノートPCで計測をした。128×128以上の解像度ではDFSの計算時間が長すぎて結果が得られなかったため、最適解は不明である。ただし、128×128の解像度では、モンテカルロ木探索もビームサーチ法も同じ最小値Iが得られており、最適解と思われる。256×256と512×512の解像度の場合、最終的に得られた最小値の値Iはビームサーチ法の方が小さかった。レイヤー数に関しても、モンテカルロ木探索で得られた最小のレイヤー数は17層で、ビームサーチ法が発見できた16層のレイヤー数のケースを見つけないことができなかった。計算時間の長さの違いがあるとはいえ、ビームサーチ法の方がモンテカルロ木探索よりも信頼性が高いものと考えられる。なお、512×512の場合、共にメモリ不足で途中終了したため、終了する前に得られた最善の最小値Iでの評価を行った。

表1. 検索時間の比較. 単位は秒. 括弧でくくられた項目は, 開始からメモリ不足により終了した時点までの時間




	 2色 64×64 Bayer 配列	 8色 48×48 最近接色	 8色 64×64 最近接色
グループ数	1534	209	311
レイヤー数	13	13	13
DFS	1.537	16354.5	1233150
BFS	1.926	(24.245)	(23.97)
モンテカルロ	456.606	355.61	207.451
ビームサーチ	4.370	4.898	19.891

表2. 大きな解像度での検索時間. 全て8色の最近接色ディザ. なお, 512×512のレイヤー数はビームサーチ法での最善解.

	64× 64	128× 128	256× 256	512× 512
グループ数	311	1034	3254	9684
レイヤー数	13	13	15	16*
モンテカルロ	105.3	758.2	611.3min	(726.6min)
ビームサーチ	11.5	3149.1	2984.2min	(5649.2min)

図10は, 物理レイヤー数を削減した場合の削減できたレイヤーのモデルである. 解像度は48×48を用いた. 一番下から4層のレイヤーと下から7,8番目のレイヤーを統合することができた. 底面の厚みとしては, 13層から9層に減ったので, 31%削減できたことになる. 一番底の4レイヤーを統合したパターンはきれいとは言えず, 黄色とシアンとマゼンタのレイヤーの一部は入り組んだ細い形状となった. これは折れやすく改善が必要である.

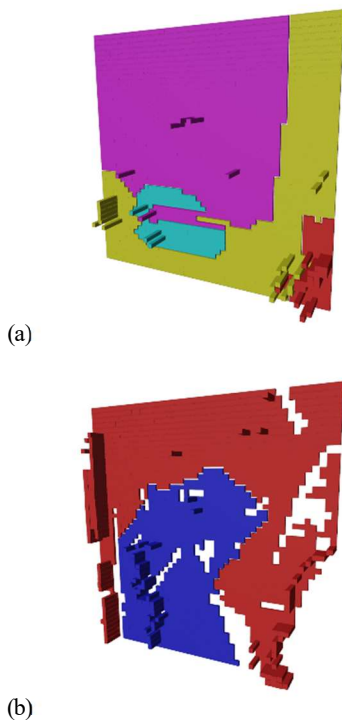


図10. レイヤーの統合. (a)下から4つのレイヤー, (b) 下から7,8番目のレイヤー

4.2 江の島の地形模型

図11は, NICOGRAPH 2023で展示した, 江の島周辺の画像から生成した地形模型のポリゴンモデルと3Dプリンターでの印刷物である. 64×64の解像度で計算を行った. 透明な水の領域を抜くために, 透明部分を一番下のレイヤーとした. モデルの起伏は, 高さマップを補間して滑らかな表面とした(付録). このモデルの構築に使用したデータが図12である. 図12(a)は, 国土地理院撮影の空中写真[12]を加工した色情報のデータである. 不透明部分の各画素の色は, 3Dプリンターのフィラメントとして用意できた8色のカラーパレットの色の内, L*a*b*色空間で最も近い色を設定した. 図12(b)は, デジタル標高地形図[13]による高さデータである. 図12(c)は, 透明部分を表すマスク画像(白い領域が透明)である. マスク画像は, 透明部分を正確に抽出するための情報として導入した. 図12(c)は, Adobe Photoshopを用いて図12(a)から作成した. レイヤー数を抑えるために不透明部分を単連結の領域とした. 今回のモデルでは, レイヤー数は18層のレイヤーとなった. レイヤーを計算するためのグループ数は152であり, 探索時間はモンテカルロ木探索で469.557秒, エニータイムビームサーチ法で30.843秒となった. このモデルの計算では, エニータイムビームサーチ法の方がモンテカルロ木探索よりも良い最適解の候補が得られた.

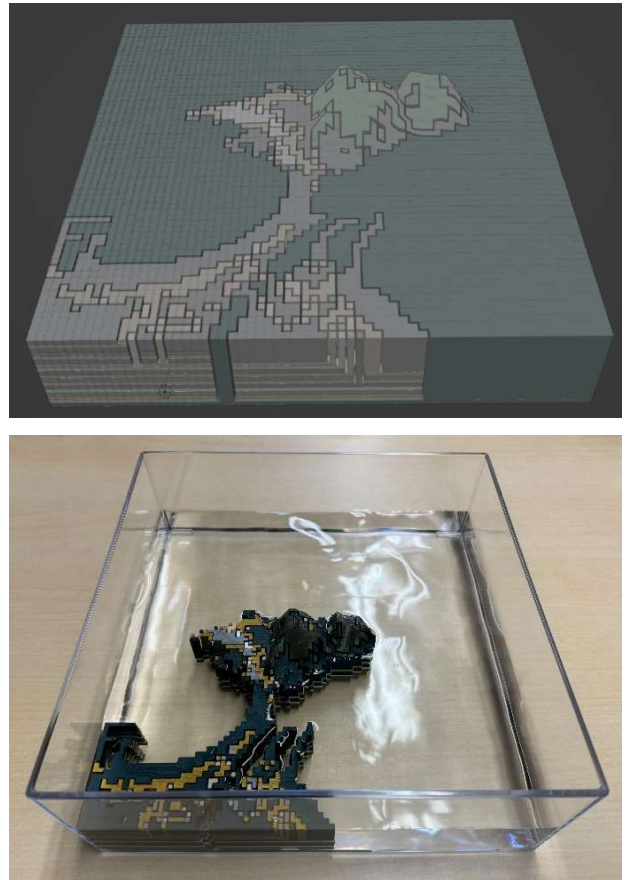


図11. 江ノ島の画像から作成した3Dモデル(上)と印刷結果(下).

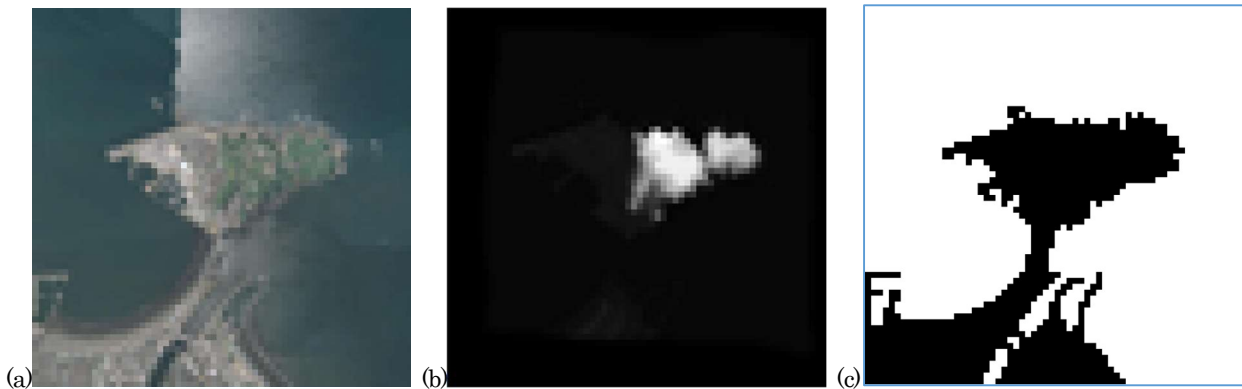


図 12. 江ノ島のモデル作成用データ. (a)色情報, (b)高さ情報, (c)透明領域を指定するマスク情報. 国土地理院の物を加工.

5. まとめ

単色の3Dプリンターの印刷物を下から積み重ねて、凹凸のある絵を作成する際の探索アルゴリズムの説明とその評価を行った。また、重ねるレイヤーの制御を行う手法を提案した。近似解の導出となるが、モンテカルロ木探索やエニータムビームサーチ法を適用することができ、両手法とも十分に高速で、3Dプリンターで印刷する印刷物の数は共に深さ優先探索で求められる正確な計算と同じレイヤー数が得られた。共に計算の幅を広げたり、計算を打ち切るルールの変更で計算時間や正確性は変化するが、本実験の範囲ではエニータムビームサーチ法の方が、より信頼性が高いという見識が得られた。

今回の手法では、大きな部品が上に来るように最適化関数を設計したが、遠く離れた場所に小さな点が存在するような部品は上のレイヤーの部品に挿入するのが難しくなるため、別の意味で組み立てやすさが変わる可能性がある。また、細い柱の形状が生じやすく、それらは折れやすいという問題がある。最適化関数の変更によりできた形状の違いによる組み立てやすさのユーザー評価は今後の課題である。また、透明部分に関して、異なる透過色のレイヤーが重なるのは好ましくない。完全に可能ではない場合も発生するが、透明部分に関しては上から見て重ならない構造を構築したい。底面の共有でも、より適切な底面の形状の研究が必要と思われる。

なお、本稿はNICOGRAPH 2023で発表した内容[16]に3.4節のレイヤーの統合及び、その結果の研究等を追記したものである。

5. 謝辞

本研究は、「色の国際科学芸術研究センター」の助成を受けたものです。

6. 参考文献

[1] 今給黎, 原, 積上げ式凹凸マップによるデザインングの立体化, NICOGRAPH 2020, E-1.
 [2] B. Taylor, A. Dey, D. Siewiorek and A. Smailagic, Customizable 3D Printed Tactile Maps as Interactive Overlays, Proceedings of the 18th International ACM SIGACCESS Conference on Computers and

Accessibility, pp. 71-79, 2016.

[3] F. Auricchio, A. Greco, G. Alaimo, V. Giacometti, S. Marconi and V. Mauri, 3D Printing Technology for Buildings' Accessibility: The Tactile Map for MTE Museum in Pavia, Journal of Civil Engineering and Architecture, Vol 11, pp. 736-747, 2017.
 [4] R. Urbas, M. Pivar, U. S. Elesini, Development of tactile floor plan for the blind and the visually impaired by 3D printing technique, Journal of Graphic Engineering and Design Vol. 7 No. 1, pp. 19-26, 2016.
 [5] C. Brown and A. Hurst, VizTouch: automatically generated tactile visualizations of coordinate spaces. In Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction, ACM, pp. 131-138, 2012.
 [6] S. Q. Xin, C. F. Lai, C. W. Fu, T. T. Wong, Y. He, and D. Cohen-Or. Making Burr Puzzles from 3D Models, ACM Trans. on Graph. (SIGGRAPH), Vol. 30, No. 4, pp. 97:1-97:8, 2011.
 [7] P. Song, B. Deng, Z. Wang, Z. Dong, W. Li, C. W. Fu, and L. Liu, CofiFab: Coarse-to-Fine Fabrication of Large 3D Objects, ACM Trans. on Graph. (SIGGRAPH), Vol. 35, no. 4, pp. 45:1-45:11, 2016.
 [8] R. Chen, Z. Wang, P. Song and B. Bickel, Computational Design of High-level Interlocking Puzzles, ACM Transactions on Graphics (SIGGRAPH 2022), Vol. 41, No. 4, pp. 150:1 -- 150:15, 2022.
 [9] R. Coulom, Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search, Computers and Games, 5th International Conference, CG 2006, pp. 72-83, 2007.
 [10] L. Kocsis nad C. Szepesvári, Bandit based Monte-Carlo Planning, In Proceedings. Lecture Notes in Computer Science. Vol. 4212. Springer. pp. 282-293, 2006.
 [11] W. Zhang, Complete Anytime Beam Search, In Proceedings of AAAI-98, pp. 425-430, 1998.
 [12] 国土地理院, 国土地理院撮影の空中写真 (2019年撮影) .
 [13] 国土地理院, デジタル標高地形図 (2019年撮影) .
 [14] E. W. Dijkstra A note on two problems in connexion with graphs. In Numerische Mathematik, 1, pp.269-271, 1959.
 [15] B. A. Galler and M. J. Fischer, An improved equivalence algorithm. Communications of the ACM. Vol. 7 No. 5, pp.301-303.
 [16] 今給黎, 原, 積上げ式凹凸マップの構築と制御に関する研究, NICOGRAPH 2023, F-3.

付録

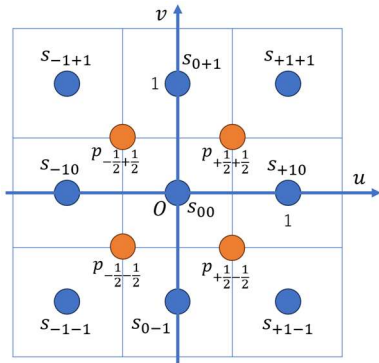
江ノ島のモデルの表面は、高さマップのテクセル中心とモデルの頂点の位置がずれている（モデルの頂点はテクセル中心間の midpoint に位置する）。そのため、高さマップの高さを補間してモデルの頂点位置としている。描画しようとしている画素のテクセル s_{00} の位置を原点とし、テクセル間距離を1とする座標系（図A）を考える。補間のための情報として縦横3点のテクセルを用いる。9個のテクセルの情報を用いて、曲面を定義する。曲面の境界条件として、画素をずらした際に連続的につながるような制約を設ける。この場合、曲面の軸を u および v とすると、次の形の曲面 p_{uv} が得られる。

$$p_{uv} = \frac{1}{2} \left(v - \frac{1}{2} \right)^2 p_{u-1} + \left(\frac{3}{4} - v^2 \right) p_{u0} + \frac{1}{2} \left(v + \frac{1}{2} \right)^2 p_{u+1}, \quad (4)$$

$$p_{ui} = \frac{1}{2} \left(u - \frac{1}{2} \right)^2 s_{-1i} + \left(\frac{3}{4} - u^2 \right) s_{0i} + \frac{1}{2} \left(u + \frac{1}{2} \right)^2 s_{+1i}, \quad (5)$$

$i = -1, 0, +1.$

モデルの頂点は、この曲面における $(\pm 0.5, \pm 0.5)$ （図Aの橙色）での値を高さとすることで、なめらかな表面を実現する。



図A. 頂点位置の補間の定義域。青丸はテクセル中心。
 橙色はポリゴンモデルの頂点位置

今給黎 隆



2009年，東京大学大学院新領域創成科学研究科修了．博士（科学）．株式会社タムソフト，株式会社ナムコ，グリー株式会社，株式会社セガを経て，2016年より東京工芸大学芸術学部ゲーム学科准教授，2021年より同教授．CG，ゲーム技術の研究に従事．芸術科学会，情報処理学会，ACM，IEEE，日本デジタルゲーム学会会員．

原 寛徳



1998年，慶應義塾大学工学部機械工学科卒業．2000年，慶應義塾大学大学院理工学研究科前期博士課程修了．2006年，慶應義塾大学大学院理工学研究科後期博士課程単位取得退学．修士（工学）．2007年に東京工芸大学に着任．芸術科学会，情報処理学会，日本デジタルゲーム学会会員．