制約付き積み上げ式凹凸マップ
Constrained Stacked Relief Maps

今給黎 隆, 原 寛徳*
Takashi Imagire, Hironori Hara*

東京工芸大学、164-8678　東京都中野区本町2－9－5
2-9-5 Honcho, Nakano-ku, Tokyo 164-8678, Japan

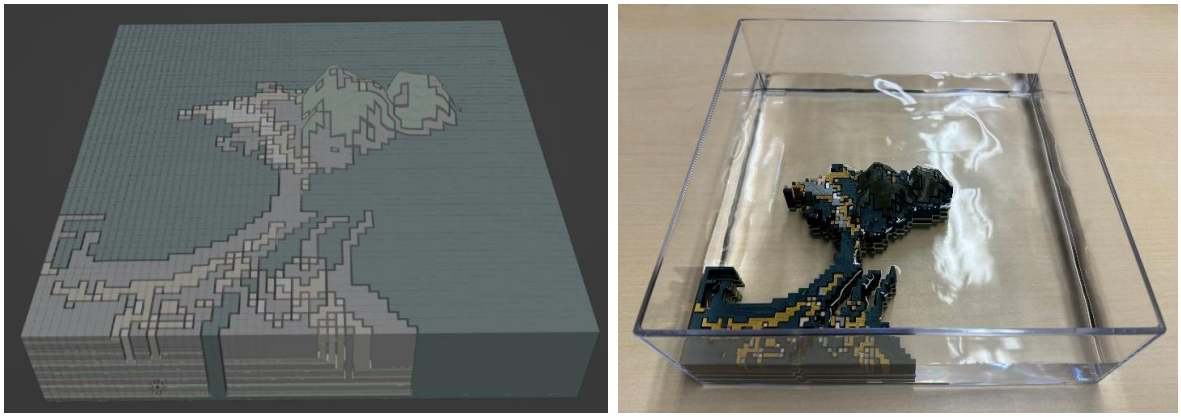


Fig. 1 江ノ島の航空写真から作成した3Dモデル(左)と3Dプリンターでの印刷結果(右)

概要

本研究は、積上げ式凹凸マップの実装及び拡張に関する研究である。図1は、18cm平方の平面を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である。この3Dモデルは画像から生成しているが、解像度が高くなったり、絵の模様が複雑になると、モデルの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。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やモンテカルロ木探索によって、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、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。

1.　はじめに

NICOGRAPH 2020 で、凹凸のある絵に関する作品を発表した[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。この作品は、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始めた。3D プリンターで印刷した部品を下から上に積み上げることによって作品が出来上がる。この作品を制作するための挑戦は、重ねるオブジェクトの個数を減らすことである。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。また、この作品の発展として電飾が考えられる。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るであろう。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。このような部品間の上下の指定は制約付き最適化問題となり、NICOGRAPH 2020 での計算よりも複雑な問題となる。

2.　提案手法

2.1 制約のない形式の解法

今回は、複数に分割した部品を、特定の順番で積み上げることで一枚の凹凸のある絵を構築する。ここで、分割した部品の数はなるべく小さくしたい。本問題を解くために画像からグラフ構造を構築する。画像について繋がっている同じ色の画素を一塊のグループと見なす。そして、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、グループ内の画素について隣接する画素があるグループの間を接続する。図2がグラフの一例である。図2(a) を変換元の画像とする。グループを計算したものが図2(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2(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。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2(d)がノードのレイヤー分けの結果である。図2(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

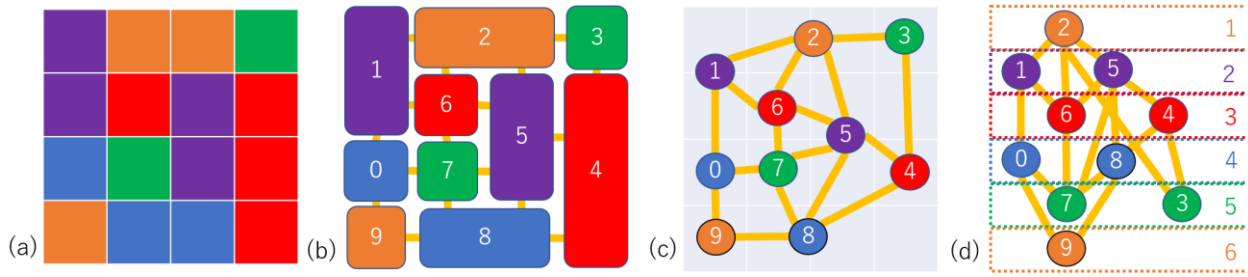


Fig. 2 画像のグラフ化と並び替え. (a)元の画像. (b)グループ化と接続情報. (c)グラフ化. (d)レイヤー分け

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。具体的には、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んで繋がっている必要がある。これを満たすには、各レイヤーのノードがそのレイヤーよりも上のレイヤーのノードを伝って到達できれば良い。

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、組み立て易さを考慮した最適化関数を導入する。小さな部品を上に置くのは大きな部品を上に置くよりも細かな作業となり、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に倒れたり、紛失しやすくなる。このため、表から見て、大きな部分ほど手前に来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を最小化する解を最適解とする。この目的関数の最適解を探すには、幅優先探索(BFS)や深さ優先探索(DFS)といった探索方法を利用することができる。

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

2.2 制約付き問題の定式化

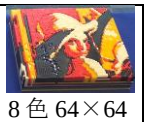
透明な部品を使う際は、その下に不透明な部品が来ては困るので、一番下の色を透明な色としたい。複数の透過色を用いる場合は、複数の透過色のレイヤーの上下関係を指定することが考えられる。この状況に対応するためにグラフを無向グラフから有向グラフに拡張する。下のレイヤーにしたい色を持つノードにつながるエッジに対して、下のレイヤーの色のノードへ進む向きに有向辺の向きを設定する。レイヤー分けの際に、有向辺の向きが下のレイヤーから上のレイヤーに向かう矢印が現れたら、そのレイヤーの色の組み合わせは無効な状況として除外する。この拡張によって、透明なノードの下に不透明のノードが入らない構造を構築できる。

ただし、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。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っ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。探索するノードがないにも関わらず、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を一つ選択し、新たな最上位ノードとして探索をやり直すことで、最後まで探索を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、ポリゴンモデルを作成する際も、最初の最上位ノードから始まるレイヤー群の高さは、別の最上位ノードから始まるレイヤー群の高さを差し引いて各表面の高さを設定すれば良い。

3. 結果

表 1 は、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。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、シンプルな DFS や BFS が高速な結果となっている。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イムビームサーチ法では、エニータイムビームサーチ法の方が 10 倍から 100 倍程度高速であるが、共に現実的な時間で計算を終了できている。

表1. 検索時間（秒）の比較。括弧の場合は、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

4。 参考文献

[1] 今給黎 隆, 原 寛徳, “積上げ式凹凸マップによるディザリングの立体化,” NICOGRAPH 2020, E-1.