立体地図模型上への道路印刷システムの開発

泉 芽衣* 渡辺 哲也*

概要. 従来，立体地図模型への道路印刷は手作業に依存しており，精度が低く多大な時間を要するといった課題があった．そこで，本研究ではこの課題を解決するため，6 軸のロボットアームに搭載された 3D ペンで印刷することで，人間の介入を最小限に抑えた印刷システムを開発した．これにより，立体地図模型上に高精度な道路印刷を自動で行うことが可能となる．

1 はじめに

現在，視覚障害者支援の分野において，触覚による情報取得を可能とする立体模型の提供が不可欠な要素となっている．その中でも地図模型は，道路や建物の配置などの空間情報を把握する上で極めて有用である．

しかしながら，これらの地図模型に道路情報を付加する製作工程には，課題が残されている．従来，立体地図模型の表面に道路を表現する際には作業者が手動で作製する手法が用いられていた．この手法は，多大な時間と労力を要する．

そこで本研究では，上記のような従来の課題，すなわち「製作時間の長期化」と「印刷精度の低さ」を同時に解決するため，ロボットアームに搭載した 3D ペンを用いた自動非平面印刷システムを開発する．非平面印刷技術自体は，電子回路形成の分野[1]でこれまでも研究されてきたが，地図模型のような複雑な自由曲面を持つ大型の対象物に対し，道路情報という高精度な線状構造を付加することに特化した独自の自動システムは確立されていない．本システムは，地図模型の三次元形状に沿って 3D ペンを高精度に制御する技術を開発する．本研究を通じて，地図模型製作における新たな自動化手法を確立し，視覚障害者が求める触覚模型を迅速に提供することを目指す．

2 開発システム

本システム（図 1）は，6 軸ロボットアームである MyCobot280, 3D ペン, および PC から構成される．

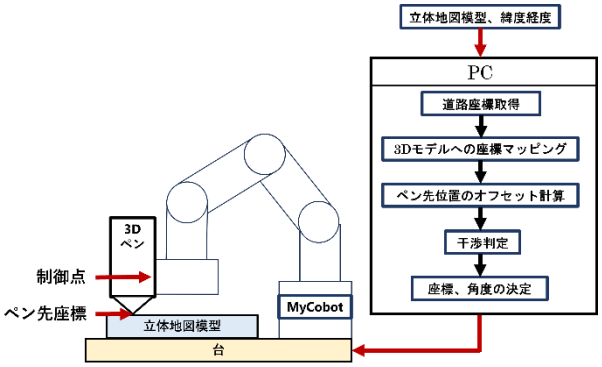

図 1. システム構成

2.1 道路の 3 次元座標取得

本システムでは，まず入力された左上と右下の 2 組の緯度経度の矩形範囲に含まれる名前のある道路データを OpenStreetMap から取得し，ユーザーが印刷したい道路を指定することで，道路の平面座標を取得する．

次に，この平面座標を立体地図模型の表面にマッピングすることで，道路の 3 次元座標を算出する．具体的には，STL ファイルから立体地図模型の寸法と表面の頂点座標を抽出し，まず，入力された 2 点の緯度経度の最大値と最小値を，STL ファイルから抽出された立体地図模型の頂点座標の平面上の最大値，最小値に変換することで，地理座標系から模型座標系へのスケーリングと位置合わせを行う．この際，SciPy の KD-Tree モジュールを用いて，平面座標から近い模型表面上の 3 つの頂点を探索する．これら 3 頂点を $\boldsymbol{P}_1(x_1, y_1, z_1)$，$\boldsymbol{P}_2(x_2, y_2, z_2)$，$\boldsymbol{P}_3(x_3, y_3, z_3)$とする．得られた 3 点から,模型表面の傾きを近似する局所的な平面の方程式を算出する．平面方程式は以下の形式で表される．

$$\mathrm{A}x + \mathrm{B}y + \mathrm{C}z + \mathrm{D} = 0 \qquad (1)$$

ここで，A,B,C,D は 3 点$\boldsymbol{P}_1$，$\boldsymbol{P}_2$，$\boldsymbol{P}_3$の座標から一意に決定される係数である．道路の高さ（z座標）は，この方程式をzについて解き，対象とする道路座標の平面座標（x, y）を代入することで決定される．

* 新潟大学

これにより，模型の表面形状に対応した道路の 3 次元座標が取得される．

2.2 制御点の取得

前節までに取得したデータは，3D ペンのペン先座標であり，実際のロボットアームの動作指令として必要なのは，MyCobot280 のアーム先端にあたる制御点である．そのため，道路座標に対して，制御点に変換されるように，ペン先からロボットアーム先端までのオフセット量をすべての道路座標に対して足し，制御点を取得した．

2.3 3D ペンと地図模型の干渉判定

本システムでは，3D ペンが地図模型の局面に沿って印刷を行う際，基本的にペンが平面に対して垂直を維持して制御させる．しかし，地形が谷状に急に変化している箇所において，ペン本体と地図模型の表面が干渉する可能性が考えられる．この干渉の有無を事前に検証し，干渉する箇所では，ペン先を支点として傾けて制御させた．

まず，Python ライブラリの Open3D を用いて，生成した軌跡の全体像とペン先の追従性を可視化した．例として，新潟県の弥彦山の模型に，弥彦山スカイラインの軌跡を生成した場合の可視化結果を示す（図 2）．

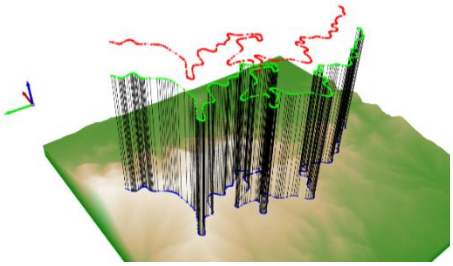

図 2. 立体地図模型と道路の可視化．赤：制御点の軌道．青：ペン先の軌道．緑：ペン固定部の軌道．黒：ペンの一部分

次に，地図模型と 3D ペンの干渉の発生箇所を特定するため，3D ペンを模した STL データを作成し，すべてのペン先座標に 3D ペンのペン先が合うように配置し，干渉の判定を行った．この際，ペン先と模型との間に，印刷に必要な隙間として 1.5mm の隙間を意図的に設けた．干渉の判定は地図模型のすべての頂点群に対して KD-Tree を用いて，地図模型表面上の最も近い頂点を探索させた．この探索結果から，3D ペンの全頂点の座標群が，地図模型の最も近い頂点の座標よりも低い位置にある場合，3D ペンが地図模型の表面にめり込んでいる状態と定義した．この定義に基づき干渉判定を実行し，干渉が検出された箇所を記録した（図 3）．

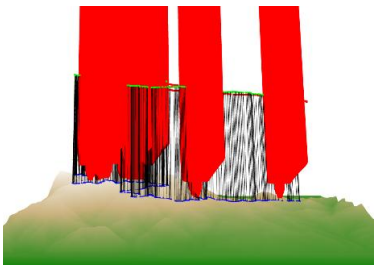

図 3. 立体地図模型と 3D ペンの干渉判定結果．赤いペンが表示されている部分が，干渉部分となる．

2.4 3D ペンの角度制御

前節で，干渉が起きた箇所のみに角度制御を設け，ペンを傾けた．まず，その点での法線ベクトルを計算した．具体的には，干渉の起きた部分の点に近い 10 点の STL 表面頂点を探索し，これらの頂点に基づき最小二乗法を用いて模型表面の局所的な平面を推定し，この平面のベクトルを法線ベクトルと定義した．そこから 3D ペン先を原点としたときのロール，ピッチ，ヨーの目標角度を算出した．この角度データに対し，前後 10 点平滑化処理を施した．これにより，より滑らかで 3D ペンが干渉しない軌道を出力するシステムを作成した（図 4）．

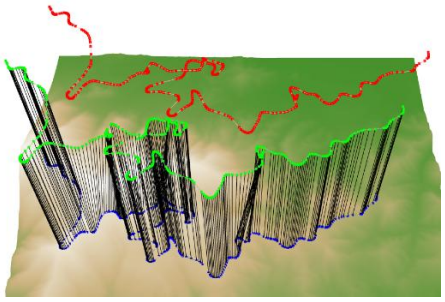

図 4. 最終的な軌道．

3 実装

制御の実装には，本来 3D ペンを取り付ける必要があるが，今回は通常のペンを簡易的に取り付けて制御確認を行った．その結果，ペンと模型との干渉は発生しなかったが，ペンの太さが異なるため，干渉の回避を検証するには不十分であった．また，同時に模型表面とペン先との間に数 mm 程度の不必要な隙間が生じることがあった．

今後は，本来の 3D ペンを取り付けて干渉回避の有効性を確認するとともに，ペン先と模型との適切な距離を保つための制御処理を導入する必要がある．

参考文献

[1] Gabriele Maria Fortunato, Matteo Nicoletta, Elisa Batoni, Giovanni Vozzi, and Carmelo De Maria. A fully automatic non-planar slicing algorithm for the additive manufacturing of complex geometries. Additive Manufacturing, vol.69, 103541, 2023.