ダイクストラ法を用いた、ポリゴン上の最短経路検索

Ito Takeru

2023 年 6 月 10 日

1　概要

今回の実験は、PLY ファイルとして与えられたポリゴンデータから、頂点とエッジの情報を取り出し、ダイクストラ法にかけることでポリゴンの表面上で疑似的な最短経路計算を行うというものである。

2　原理

以下に今回の実験で用いられる用語について解説する。

ダイクストラ法とは、重みのついたグラフ上の最短経路問題を解くためのアルゴリズムである (例: 図 1 参照)。

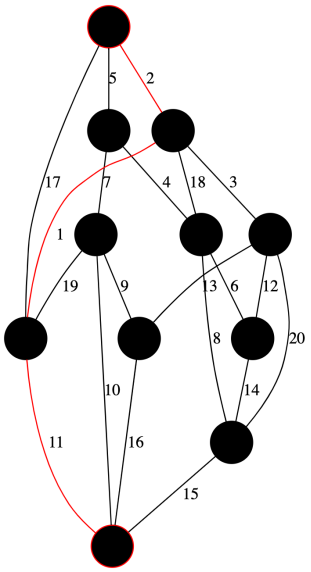

図 1　ダイクストラ法を適用した例

次に PLY ファイルについて述べる

PLY (Python Lex-Yacc) は、3D モデリングソフトウェアなどで使用されるファイルフォーマットである [1]。PLY ファイルは、頂点と面の情報の情報が入力されており、CAD や meshlab などの 3D オブジェクトを扱うツールで読み込むことができる。

3　実験手順

3.1　最短経路の描画

まず、PLY ファイルとして web サイト上で公開されていた ply ファイル、helix.ply を利用した (図 2)

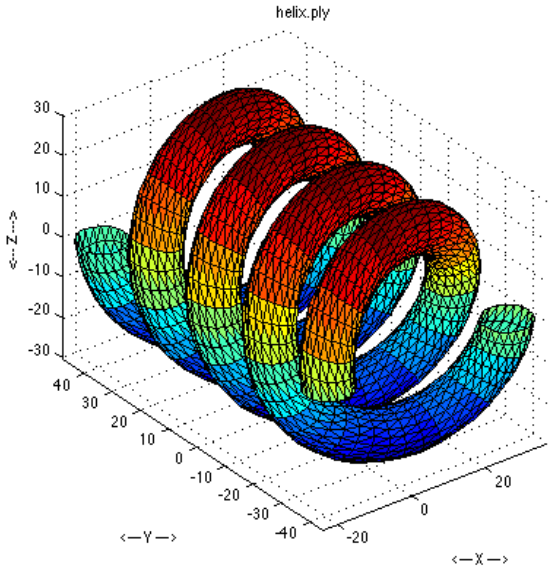

図 2　helix

次に Python を用いてこのファイルからノードとして頂点情報を、エッジとしてポリゴンの三角形の辺の情報を抜き出し、エッジの重みとして辺の長さを使うことでグラフの情報を取り出した。そして、できたグラフデータのノードの内 2 つを、スタートとゴールのノードに指定し、C 言語のダイクストラ法のアルゴリズムをを実行し、重みの最も少ないノードの順序を求めた。

3.2　隣接行列と優先度キューによるダイクストラ法の実行速度比較

同様に [2] において公開されている 3 つの PLY データ airplane, teapot, apple(図**??**) において、全ての点の組間で、隣接行列に基づいたダイクストラ法と優先度キューに基づいたダイクストラ法を実施したときの計算量を比較した。また、それぞれの図形のノード、エッジ数は表 1 の通りであり、計算量はプログラムの繰り返し処理ごとに +1 されるような変数を用意してカウントした。

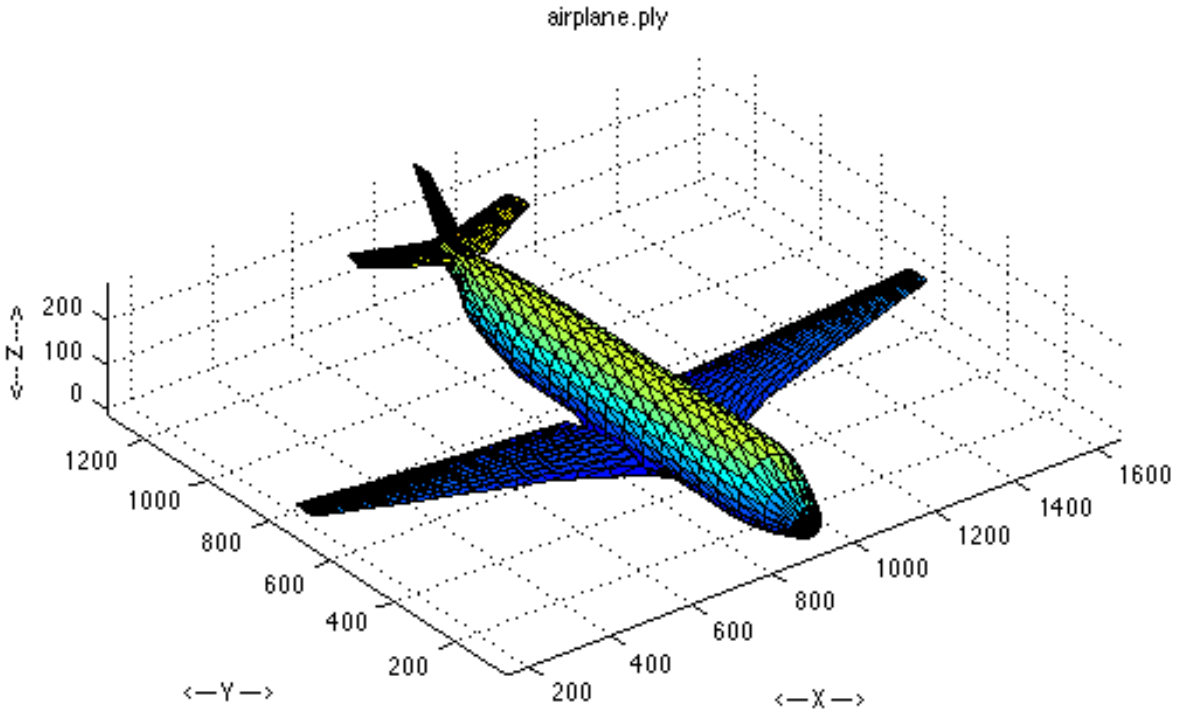

図 3　airplane の図

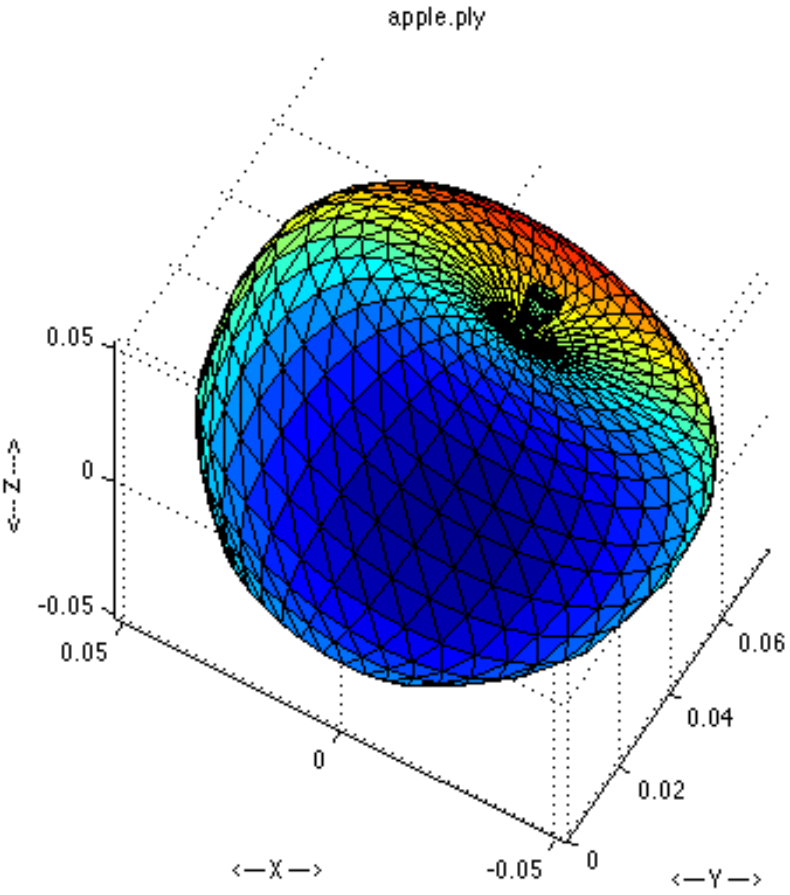

図 4　apple の図

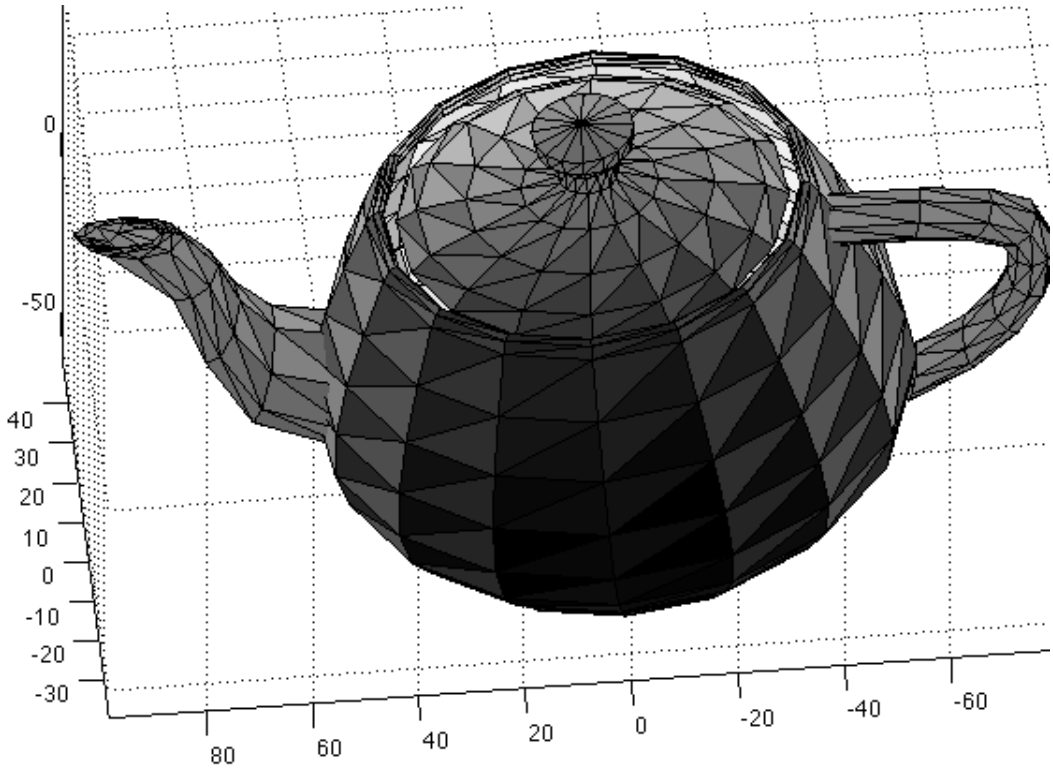

図 5　teapot の図

図形	ノード数	エッジ数
airplane	1335	3789
apple	867	2568
teapot	1177	3432

表 1　3 つの図形のノード、エッジのステータス

4　結果

4.1　最短経路の描画

ダイクストラ法を実装した結果を、Python を用いて可視化すると以下のようなことが分かった。図 8, 9 を見ると点は螺旋の内側を通っており、正しく最短に近い経路を取っていることが分かった。しかし、図 9 に良く表れているように、検出した経路に折れ曲がりが見られ、ポリゴン表面上における本当の最短経路ではないことがわかる。

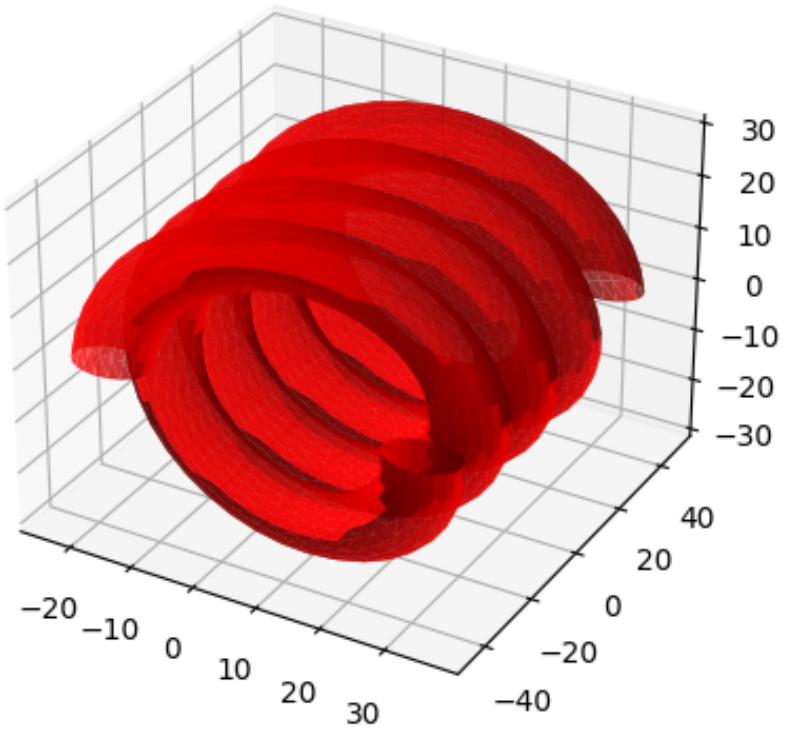

図 6　元の helix

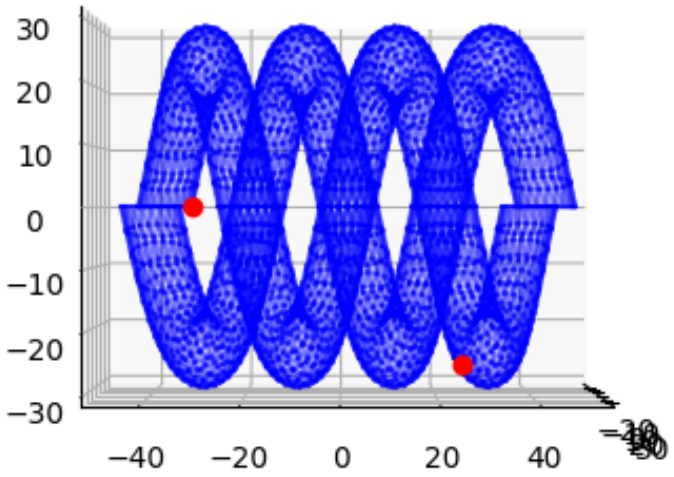

図７　ノードとエッジに分解されたポリゴンと設定した始点と終点

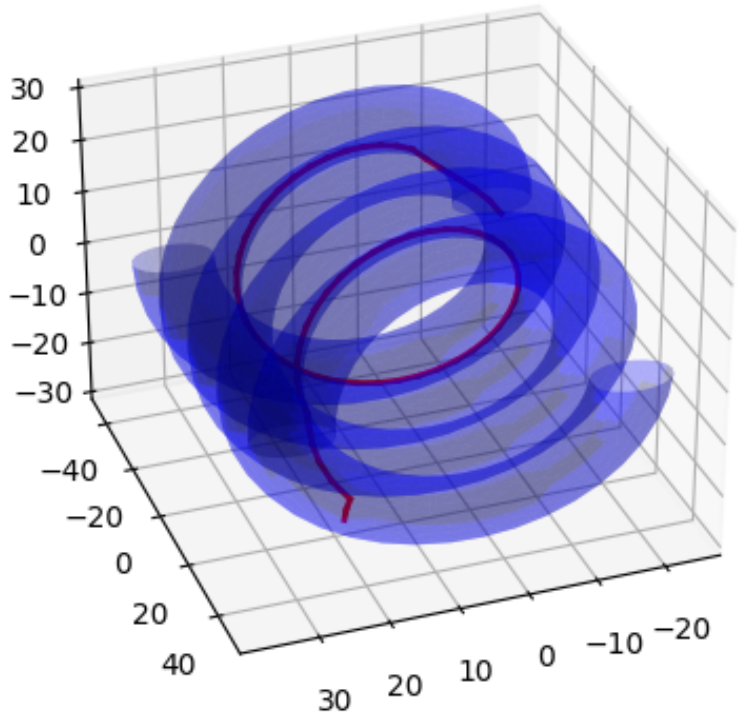

図 8　ダイクストラ法によって求めた最短経路

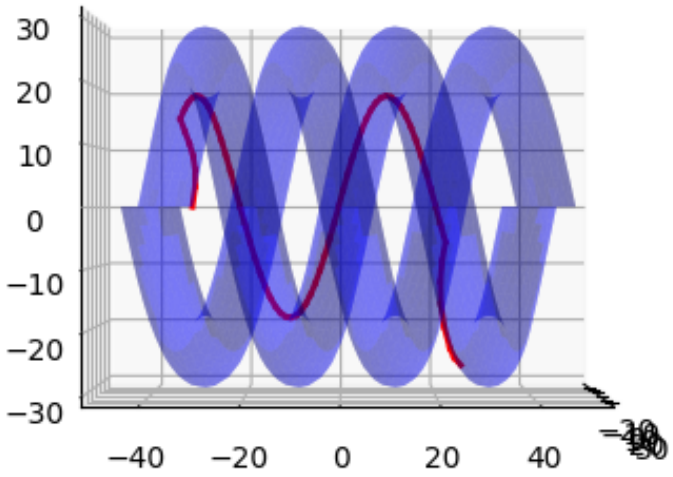

図 9　ダイクストラ法によって求めた最短経路

4.2　隣接行列と優先度キューを用いたダイクストラ法の比較

隣接行列を用いたダイクストラ法と、優先度キューを用いたダイクストラ法をグラフの全ての点間で実施したときの、1 組の点間におけるダイクストラ法の平均計算量は以下のようになった。

図形	ノード数	エッジ数	dijkstra(優先度キュー)	dijkstra(隣接行列)
apple	867	2568	35523	641016
airplane	1335	3789	23101	434551
teapot	1177	3432	30956	648059

表 2　図形とグラフ構造ごとの計算量比較

5　考察

最短経路の可視化を通してわかったことを述べる。まず、この最短経路は通れる場所が頂点とエッジのみであることから、表面全体における最短経路とは異なった所を通ることが予想できるが、今回の例で行った範囲で考えると、かなり表面全体における最短経路に近い経路を通ることが

分かった。また、今後の展望としては以下のようなものが考えられた。

- ポリゴンをどのように細分化していったら、ダイクストラ法の結果は真の最短経路に収束するのか
- 曲面上に幾何学的なパターンのポリゴンを構成することで幾何の接続のような概念を疑似的に表せないか

隣接行列と優先度キューのダイクストラ法アルゴリズムを比較して考えたことをのべる。表を見ると、apple はノード、エッジ数が他の図形に比べて少ないにもかかわらず airplane よりも多く、teapot と同程度の計算量を要した。これは、図形の連結性に原因があり、teapot は蓋と容器、airplane は翼と胴体は繋がっておらず、そのためこれらのダイクストラ法のアルゴリズムは途中で打ち切られたことが原因だと考えられる。特に、連結していないことの判定において、ある点に対して暫定距離が最小のノードを取り出す際に、優先度キューならば root のノードの暫定距離が INTMAX であること、隣接行列なら、visited ノードが true なノードの全てを確認しその中のノードが全ての距離が確定済であること、を条件にする。このことから、優先度キューのほうが連結でないことをより少ない計算量で判定することができると考えられる。これが apple における優先度キューのダイクストラ法計算量が、他の図形の優先度キューに比べて大きいという結果に表れたと考えられる。

全ての図形において、優先度キューのほうが隣接行列よりも計算量が小さかった。これは、ポリゴンが成立する条件により、グラフが常に疎行列であるからだと考えられる。また、1 つのノードに繋がっているエッジの数を n とすると、それぞれの計算量に以下の関係が成り立ち、このことも結果を裏づけていると言える。まず以下が成立。

$$V \approx \frac{2E}{n} \tag{1}$$

よって、隣接リストの計算量は

$$O((V+E)\log V) \approx O(V \log V) \tag{2}$$

そして、隣接行列の計算量は

$$O(V^2 + E) \approx O(V^2) \tag{3}$$

このことからポリゴン上のダイクストラ法において、一般的に隣接リストの方が計算量が少ないことが分かる。

参考文献

[1] Paul Bourke. PLY - Polygon File Format. http://paulbourke.net/dataformats/ply/

[2] FSU. PLY Files an ASCII Polygon Format https://people.sc.fsu.edu/~jburkardt/data/ply/ply.html