TriangularMeshTransformation

Quiz

半数以上の人にとっては、あくまでも「余暇」のためのクイズです。

This is just for your leisure quiz, if your research topic is not directly related to this.

Refresh your memory on the quiz on April 2nd as below.

There must be multiple ways to handle it.

I have some soggestions to some members based on their project.

Therefore the members who are nominated below should check the corresponding section.

- A solution version for Okada-takuya san will appear later
- A solution version to which Wada-takuya san might be most ready will appear below the Oka-taku version.
- A solution version for Yusri-san follows. However Yusri-san version is somewhat premature. Why I say this is related to Yusri-san? It is because the idea is related with Frenet-Serre not for a curved line but for a curved surface.

SUper-elementary solution (1) べたべたな解き方、その１

Let $|V|$ denote the number of vertices. 3D coordinates of the all vertices are consisted of $3 \times |V|$ numbers. We care two sets of $3 \times |V|$ number sets.

頂点数 $|V|$としたときに、全部で$3 \times |V|$の座標値のペアがある。

Let's call the vectors X and Y. We should find $|V| \times |V|$ matrix M that satisfy the formula below.

これを、$X = (X_1, \ldots, X_{3\times|V|})$、$Y = (y_1, \ldots, y_{|V|})$という長さ$3 \times |V|$のベクトルの線形変換として、$|V| \times |V|$行列$M$

$$Y = MX.$$

This problem is too easy or there are so many solutions. この問題は簡単すぎるというか、解がいくらでもあるのでよろしくない。

We should add restricutions to make our solution specific.

制約をつけて、解が一意になるようにするとよい。

For exapmle, we can make $|V| \times |V|$ matrix whose 1st coloumn is X and the all other columns are the same with Identity matrix. In the same way we can make $|V| \times |V|$ matrix for the vector$Y$. Then M will be specifically solvable. And the degree of freedom of this solution is $3 \times |V|$.

たとえば、$|V| \times |V|$の単位行列を作り、その第１カラムをベクトルXとし、同様に、第１カラムだけがベクトルYであるよう正方行列を作れば、Mは一意に求まる。このときMを解くための自由度は$3 \times |V|$と成っている。

This sounds trivial, but every vertex's coordinates are calculated from all vertices' coordinate values.

This point is something We want to do; in other words, we want to care all vertices should be mutually related and transformation should have such features in the calculation.

この方法はつまらないように思えるが、見方を変えるとそうでもない。どの点の座標変換も、メッシュのすべての座標の影響を受けて決まる、方式になっているから。 この点は、大事、なぜなら、僕らは、メッシュ全体を一つの対象としてとらえ、その上での変換とはどういうものかを考えたいから。

Super-elementary solution (2) べたべたなとき方、その２

Assume $|V| = 3n$. Rather than treat all triagles, we can treat 1/3 of triangles such that cover all vertices.

Then we should find 3×3 matrix that transform one triangle to its corresponding triangle.

Eventually $(3 \times 3) \times n = 3 \times |V|$ unnown variables will be obtained.

頂点数が３の倍数だとしよう。そして、すべての三角形について変換を求める代わりに、すべての頂点をカバーするにたるだけの三角形を選び出し、それらについてだけ3×3変換行列を求めるという手もある。

結局、$(3 \times 3) \times n = 3 \times |V|$の変数を確定する作業であることになる。

With this way, we can place 3×3 matrices on the surface and we have distribution of 3×3 matrices on the curved surface (or its corresponding unit sphere surface).

Okada-takuya san　岡田卓也さん

Although the original quiz was the linear transformation of multiple triangles coordinates altogether, Okada-takuya san's research topic is

オリジナルの問題は、複数の三角形の座標の線形変換問題ですが、岡田卓也さんの研究テーマは以下の諸点からなっています。

- Embed objects onto a unit sphere 対象を単位球面に埋め込み
- Decompose the 3D coordinates on the sphere into spherical harmonics 球面上に貼り付けた3D座標を球面調和関数分解し
- Using Althlooti algorithm, arrign the 3D embedding with another object so that some mesure should be minimized with quaternion-based linear algebra. Althlootiアルゴリズムを使って、２つの球面調和関数分解係数行列の最適アラインメントを四元数的な線形代数解法で取り出す

Therefore, Okada-takuya san would be better to understand the role of this spherical harmonics decomposition step in the "multi-triangle linear transformation". したがって、岡田卓也さん的には、球面調和関数分解のステップが、三角形座標変換にどういう風に関与するのか、という視点を持って、このクイズを考えるとよいと予想されます。

Wada-takuya san 和田拓也さん

Wada-takuya san is quite familiar with mesh objects. Triangles, verices, cotangent-formula, dual graphs and so on.

和田拓也さんは、メッシュオブジェクト(三角形、頂点、コタンジェントフォーミュラ、双対グラフなど)の扱いに相当習熟しています。

Therefore the following idea can be relatively easily understood.

なので、以下のような発想をすることは比較的自然でしょう。

- Pick up two mutually corresponding triangles. 対応する２三角形を取り出す
- Their circumcenter and their radius can be calculated. その三角形の外心と外接円半径とを求める
- Transformation of the 1st triagnel to 2nd triangle can be decomposed into;第一の三角形から第二の三角形への移動を次のように分解する
- i. movement of circumcenters (Cartesian/polar coordinates in 3D space), pay attention to the normal vector direction 外心の移動(デカルト座標系/極座標系。外接円の法線ベクトル方向に留意)

- ii. change in circumradius 外心半径の変化
- iii. averaged rotation angle of the vertices along the circle and find the angle that is the “average of angles of three vertices” 外接円周上の３点を平均的に説明する回転角
- iv. the deviational component of angles from the “averaged angle”. 平均回転角で定まる点からの各点のズレを表す角

I guess these information should have 9 degrees of freedom…

Why? Because it should be…

This method extract the triangle-change into multiple features, each of which are easily described. この方法だと、三角形の変化が、複数の性質に分解できて、しかも、それぞれの性質の意味が解りやすい。

Also the featuring values can be mapped on the object/sphere surface. しかも、それぞれの特徴量が形表面・球面に貼り付けられる。

The values mapped on the surface are ready to be processed with smoothing procedures. The outputs of smoothing procedures would be easier to be compared than the originals.

張り付いた意味の取りやすい値はスムージング処理の対象にしやすいので、形全体の変化をスムージングした後に比較することもできそう。

Hierarchical Decomposition 階層的分解

When we handle multiple objects as a whole, hierarchical handling works nicely in many cases. なにかたくさんのものを全体として扱うとき、階層的な取り扱いが良い結果をもたらすことは多い。

In case of triangular mesh with multiple vertices and their coordinates, how can we handle them hierarchically?

多数の頂点とその座標からなる三角メッシュが対象となっているとき、階層的な扱いとはどういうことになるだろうか？

- Weight center should be the root. 重心をそれぞれ求めその移動を取り出す
- Magnification. 拡大縮小(の平均値)
- 3D rotation to minimiza something 何かを最小化する3D回転
- Deviation from them. それ以外のズレ成分

Actually, this approach is the most simple. 実際、この方法が最も素直とも言える。

However the optimization of 3D rotation was too difficult. That is why Okada-takuya san’s approach adopted Althlooti method as its good substitution. しかしながら3D回転の最適解を求めるのが難しいので、岡田卓也さんのAlthlooti法を採用することになっています。 というわけで、この作戦は常に意識しつつ、現在は保留にします。

Moving Frame on the curved surface : Frenet-Serret

When you have a curve in 3D space, you can put a orthonormal basis on the points in the curve and evaluate the movements of the basis along the curve. This basis is called “moving frame” because the frame moves and the approach is called “Frenet-Serret method”.

Yusri-san has been using this method to evaluate trajectories of cells.

3D空間に曲線があるとき、曲線上の点に正規直交規定を置いてやり、曲線にそってその正規直交規定がどのように回転していくかを追いかけていくやり方があります。曲線状の正規直交規定を動標構と言います。またこの方法をフレネ=セレ法とも言います。 ユスリさんはこの方法を細胞の軌跡解析に使用しています。

Almost always, mathematics expands a method from one dimeion to higher dimension.

数学では、ほぼ間違いなく、１次元の方法を高次元化します。

A curve is a one dimensional manifold. 曲線は１次元多様体。

A curved surface is a two dimentional manifold. 曲面は２次元多様体。

Therefore curved surface version of “moving frame” should be defined and we should attache them on the curverd surface and we should find the way to quantitate the change of the “moving frames” that are located side-by-side.

したがって、曲面用のmoving frameを定義して、それを曲面に貼り付け、隣り合う曲面用moving frameの変化の定量法を決めればよいのではないでしょうか。

These information should determine the relative location of triangles’ vertex coordinates.

この情報が三角形の相対的な座標を定める情報を持っているのではないかと思います。

Since Frenet-Serre required second differences, curved surface version will also require second differences of something.

フレネ=セレは二階差分を必要とする方法だったので、曲面版でも、何かの２階差分を考える必要はあるでしょう。

Since every points on the curved surface, two principal curvatures can be defined and they have their own directions, they might be something to start with…

曲面上の点には２つの主曲率が定まり、それには方向もあるので、そのあたりを取っ掛かりにするとよいのでは。。。

Triangular mesh objects 三角メッシュオブジェクト

If you want to have objects in your hands, you may use the followings .もしデータセットがあった方がやりやすいようなら、以下のコードを使ってもよいでしょう。

Some packages to be used should be installed.

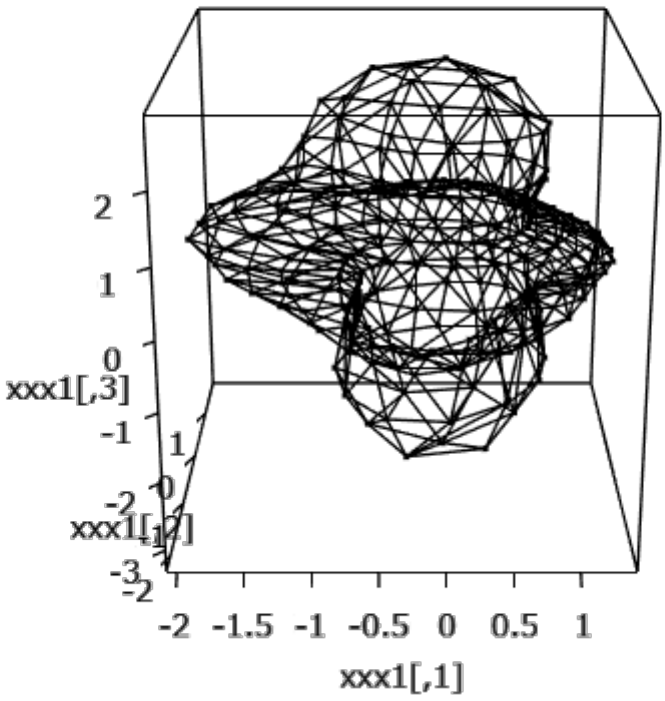
If you are not familiar with R and/or Rmd, it is a good idea to get familiar with them as soon as possible to obtain useful information from Yamada.

```
#library(devtools)
#install_github("ryamada22¥Ronlyryamada")
#install.packages("RFOC")
library(Ronlyryamada)
library(RFOC)
```

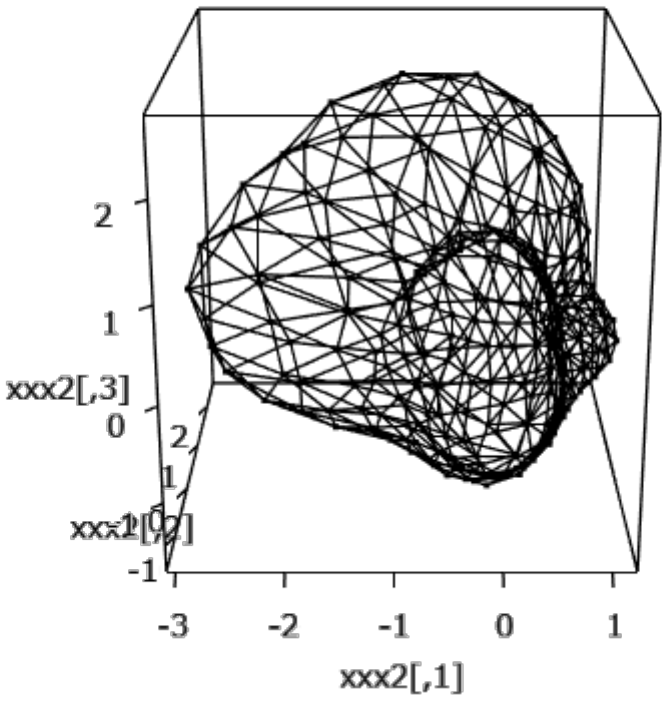
```
## Warning: package 'RFOC' was built under R version 3.4.4
```

The codes below are a bit longer than should be, but ignore the details and just believe that they make two 3d embeddings of the same triangular mesh graph.

```
k <- 5
n <- 5
n.mesh = 16
scale.shift = 0.1
scale.rotation1 = 0.1
scale.rotation2 = 0.1
scale.shear = 0.1

A. <- matrix(runif(n^2), n, n)
A.[1, 1] <- k
B <- matrix(rnorm(n^2), n, n)
xxx <- my.spherical.harm.mesh(A = A., B = B, n = n.mesh)
plot3d(xxx$v)
segments3d(xxx$v[c(t(xxx$edge)), ])
M <- diag(rep(1, 4))
M[1:3, 4] <- runif(3) * scale.shift
theta1 <- runif(1) * scale.rotation1
theta2 <- runif(1) * scale.rotation2
Mm <- diag(rep(1, 3))
Mm[1:2, 1:2] <- my.2d.rot(theta1) %*% Mm[1:2, 1:2]
Mm[2:3, 2:3] <- my.2d.rot(theta2) %*% Mm[2:3, 2:3]
M[1:3, 1:3] <- Mm
M[4, 1:3] <- rnorm(3) * scale.shear

n.step <- 30
for (i in 1:n.step) {
        A. <- A. + rnorm(n^2, 0, 0.05)
        xxx <- my.spherical.harm.mesh(A = A., n = n.mesh)
        xxxx <- cbind(xxx$v, rep(1, length(xxx$v[, 1])))
        rot.xxxx <- t(M %*% t(xxxx))
        rot.xxxx. <- rot.xxxx[, 1:3]/rot.xxxx[, 4]
}

xxx1 <- rot.xxxx.
plot3d(xxx1)
segments3d(xxx1[c(t(xxx$edge)), ])
```


2
1
0
xxx1[,3]
-1
-2
-3
1
0
-1
-2
-2 -1.5 -1 -0.5 0 0.5 1
xxx1[,1]

```
A. <- matrix(runif(n^2), n, n)
A.[1, 1] <- k
B <- matrix(rnorm(n^2), n, n)
xxx <- my.spherical.harm.mesh(A = A., B = B, n = n.mesh)
plot3d(xxx$v)
segments3d(xxx$v[c(t(xxx$edge)), ])
M <- diag(rep(1, 4))
M[1:3, 4] <- runif(3) * scale.shift
theta1 <- runif(1) * scale.rotation1
theta2 <- runif(1) * scale.rotation2
Mm <- diag(rep(1, 3))
Mm[1:2, 1:2] <- my.2d.rot(theta1) %*% Mm[1:2, 1:2]
Mm[2:3, 2:3] <- my.2d.rot(theta2) %*% Mm[2:3, 2:3]
M[1:3, 1:3] <- Mm
M[4, 1:3] <- rnorm(3) * scale.shear

n.step <- 30
for (i in 1:n.step) {
        A. <- A. + rnorm(n^2, 0, 0.05)
        xxx <- my.spherical.harm.mesh(A = A., n = n.mesh)
        xxxx <- cbind(xxx$v, rep(1, length(xxx$v[, 1])))
        rot.xxxx <- t(M %*% t(xxxx))
        rot.xxxx. <- rot.xxxx[, 1:3]/rot.xxxx[, 4]
}
plot3d(rot.xxxx.)
segments3d(rot.xxxx.[c(t(xxx$edge)), ])

xxx2 <- rot.xxxx.
plot3d(xxx2)
segments3d(xxx2[c(t(xxx$edge)), ])
```


2
1
xxx2[,3]
0
2
1
xxx2[,2]
-1
-3
-2
-1
0
1
xxx2[,1]