BestFit

Problème posé :

On dispose de 2 triangulations représentant chacune partiellement un même objet. Les 2 triangulations comportent une partie commune significative de l'objet. Chaque triangulation est définie dans 2 repères différents, la transformation permettant de passer d'un repère à l'autre est inconnue. Le problème consiste à déterminer la transformation permettant d'appliquer au mieux les parties communes des 2 triangulations l'une sur l'autre, c'est-à-dire de rendre minimale l'erreur quadratique moyenne lorsqu'on effectue la projection des points d'un modèle sur l'autre modèle.

Principe mathématique de la méthode utilisée :

On raisonne par analogie avec un modèle possédant toutes les qualités de continuité requises. Soit donc une surface S et une surface S1, une translation T(xt,yt,zt) , et une rotation R(φ,θ,ψ) qui font passer de la surface S1 à la surface S. Les 6 coefficients de la transformation sont les inconnues à déterminer.
On suppose que les angles de rotation (autour de oz, oy et ox respectivement et dans cet ordre) sont assez petits pour qu'ils soient assimilables à leur sinus, et que leur cosinus soit assimilable à 1.

Soit un point M de S1, son image M' sur la surface S est liée à M par la transformation :

$$\overrightarrow{OM'} = R.\overrightarrow{OM} + T$$

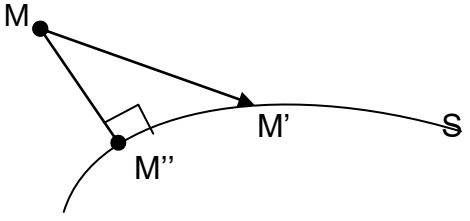
on ne connait pas M' puisque la transformation est inconnue, par contre on peut calculer la projection orthogonale M'' de M sur la surface S et définir une erreur

$$d = \overline{MM''}$$

distance algébrique du point M à la surface S.

En assimilant les angles à leur sinus, et en négligeant les termes du second ordre la transformation s'écrit sous forme matricielle :

$$R = \begin{pmatrix} 1 & -\psi & \theta \\ \varphi & 1 & -\psi \\ -\theta & \psi & 1 \end{pmatrix}$$



Calculons l'influence de la transformation sur l'erreur MM'' :

On a

$$\overrightarrow{MM'} = \overrightarrow{OM'} - \overrightarrow{OM} = (R - E).\overrightarrow{OM} + T$$

$$= \begin{pmatrix} -\varphi.y + \theta.z + xt \\ \varphi.x - \psi.z + yt \\ -\theta.x + \psi.y + zt \end{pmatrix}$$

Où x, y, z sont les coordonnées du point M

et :

$\overline{MM''} = \overrightarrow{MM'}.\vec{n}$ avec n (xn,yn,zn) normale à S passant par M

soit :

$\overline{MM''} = xn.(-\varphi.y + \theta.z + xt) + yn.(\varphi.x - \psi.z + yt) + zn.(-\theta.x + \psi.y + zt)$

En factorisant φ, θ, ψ :

$\overline{MM''} = \varphi.(-y.xn + x.yn) + \theta.(z.xn - x.zn) + \psi.(-z.yn + y.zn) + xt.xn + yt.yn .zt.zn$

En posant pour chaque point M = Mi : $di = \overline{MiMi''}$,

ai = (-yi.xni + xi.yni) bi = (z.xn – x.zn) ci = (-z.yn + y.zn) on obtient pour chaque point Mi une forme linéaire li fonction des 6 inconnues φ, θ, ψ, xt, yt, zt :

$$li = \varphi.ai + \theta.bi + \psi.ci + xt.xni + yt.yni + zt.zni - di$$

On cherche la valeur des 6 inconnues qui rendent minimales la somme des carrés des formes linéaires :

$$\Sigma_i l_i^2 \quad \text{minimal} / \varphi, \theta, \psi, xt, yt, zt$$

Ce qui est obtenu lorsque toutes les dérivée partielles / φ, θ, ψ , xt, yt, zt

sont nulles, ce qui donne un système linéaire de 6 équations à 6 inconnues à résoudre de la forme :

$AX - B = 0$ avec

$$B = (\Sigma a_i.d_i , \Sigma b_i.d_i , \Sigma c_i.d_i , \Sigma x_i.d_i , \Sigma y_i.d_i , \Sigma z_i.d_i)$$

$$\text{et } A = \begin{pmatrix} \Sigma a_i.a_i & \Sigma a_i.b_i & \Sigma a_i.c & \Sigma a_i.y_i & \Sigma a_i.z_i & \Sigma a_i.z_i \\ \Sigma b_i.a_i & \Sigma b_i.b_i & \ldots\ldots & & & \Sigma b_i.z_i \\ \Sigma c_i.a_i & \Sigma c_i.b_i & \ldots\ldots & & & \Sigma c_i.z_i \\ \Sigma xn_i.a_i & \Sigma xn_i.b_i & \ldots\ldots & & & \Sigma xn_i.z_i \\ \Sigma yn_i.a_i & \Sigma yn_i.b_i & \ldots\ldots & & & \Sigma yn_i.z_i \\ \Sigma zn_i.a_i & \Sigma zn_i.b_i & \ldots\ldots & & & \Sigma zn_i.z_i \end{pmatrix}$$

Parmi les difficultés d'application possibles :

- La linéarisation utilisée n'est valable que si les repères ne sont pas trop différents, la méthode nécessite d'effectuer des itérations. On peut ne pas converger si les écarts sont trop importants.
- On ne connait pas a priori quels sont les points du maillage 1 qui ont une projection orthogonale sur le maillage 2, tant que les 2 maillages ne sont pas dans le même repère. Si dans leur position finale théoriques, les surfaces présentent des zones en vis-à-vis qui sont par conception différentes, (zone d'une surface ne s'appliquant pas sur l'autre surface), ces zones devront être éliminées au préalable.
- Un maillage n'est pas une surface continue G1 : les dérivées intervenant ne sont pas des fonctions continues, un processus itératif, même s'il n'est pas divergeant, ne converge pas forcément vers une limite, mais risque d'osciller dans une plage de valeurs.
- Si les parties communes des 2 surfaces admettent des symétries le problème est indéterminé.
- La méthode recherchant la valeur minimale d'une fonction, peut aboutir à un minimum local qui n'est pas la solution.
- On doit disposer d'une fonction de projection efficace (temps de calcul)

Algorithme de base :

La méthode utilisée étant une approximation linéaire, l'algorithme de base doit être appelé itérativement pour obtenir la solution. Séquence d'appel :

```
int Stl_Maillage::Stl_BestFit(int numObj1,int numObj2,
                              Geo_deplacement *pDin = NULL);
```

On suppose qu'au préalable, les 2 maillages ont été chargés dans le « Stl_Maillage » , avec les identifiants numObj1 et numObj2 respectivement .

Le pointeur « `Geo_deplacement *pDin` » est un argument optionnel.

- S'il est absent ou nul, la transformation géométrique résultant de l'appel est automatiquement appliquée au maillage1 de façon qu'il se situe dans le même repère que l'objet 2 (les coordonnées des points de l'objet1 sont modifiées). « `Geo_deplacement` » est une classe décrivant les déplacements dans l'espace 3D (rotation par rapport à un axe quelconque passant par l'origine, suivi d'une translation).

- Si le déplacement est fourni en entrée, il est considéré comme une première approximation du changement de repère à effectuer pour placer l'objet1 dans le même repère que l'objet2. En sortie, la transformation géométrique calculée est uniquement appliquée au déplacement : les coordonnées de l'objet1 ne sont pas modifiées. On peut fournir en entrée un déplacement initial nul , qui est le déplacement par défaut créé par le constructeur de la classe , puis le déplacement est modifié à chaque itération :

```
Geo_deplacement *pDin = new Geo_deplacement() ;
//calcul itératif du déplacement
for( int k=0 ; k<nbiteration ; ++k)
      pMaillage->Stl_BestFit(numObj1,numObj2,pDin) ;
//application du déplacement
pMaillage->Deplace(deplacement,numObj1); //application du déplacement
```

La fonction membre « `Deplace` »'applique le déplacement à tous les points de l'objet numObj1.

La classe Geo_deplacement :

Elle définit un déplacement dans l'espace 3D (produit d'une rotation et d'une translation). Du point de vue interne, elle est définie par une matrice orthonormée et par un vecteur de translation. Elle comporte plusieurs constructeurs en particulier :

construction de l'opérateur produit des 3 rotations φ, θ, ψ et de la translation xt, yt, zt, de l'opérateur identité (déplacement nul),de l'opérateur appliquant un trièdre sur un autre trièdre, ainsi que quelques fonctions utiles : produits de 2 déplacements : si D1 et D2 sont 2 « Geo_deplacement » , l'opération

```
D1.Produit(D2)
```

Effectue le produit du déplacement D1 par D2 et remplace D1 par le résultat de l'opération. Un déplacement peut s'appliquer à un point CasCade (gp_Pnt) ou à un point Stl_Point à l'aide des fonctions membre

```
Void Geo_deplacement::Deplace(gp_Pnt &Point)

Void Geo_deplacement::Deplace(Stl_Point *pPslPnt)
```

Et le déplacement inverse à l'aide de la fonction

```
Void Geo_deplacement::DeplaceR(gp_Pnt &Point)
Void Geo_deplacement::DeplaceR(Stl_Point *pPslPnt)
```

Lorsqu'on applique la fonction « `Deplace` » à un vecteur CasCade gp_Vec , seule la matrice de rotation est appliquée, le vecteur n'est pas modifié par la partie « translation ».

La classe Stl_Triedre :

Un élément de la classe trièdre est défini par un point origine et 3 vecteurs orthonormés. Parmi ses constructeurs, la construction à partir de trois points gp_Pnt :

```
Stl_Triedre(gp_Pnt *pP1,gp_Pnt *pP2,gp_Pnt *pP3)
```

Les trois points ne jouent pas le même rôle : P1 définit l'origine du trièdre, $\overrightarrow{P1P2}$ définit la direction du premier vecteur et P3 définit avec P1 et P2 le plan contenant les 2 premiers vecteurs du trièdre. Les 3 points ne doivent pas être alignés ni confondus. Les différents constructeurs fournissent toujours un trièdre direct : si V1, V2, V3 sont les 3 vecteurs du trièdre, on a toujours V3 = V1^V2.

Une méthode interactive permettant de positionner un objet par rapport à un autre objet consiste à mettre en correspondance 3 points définis sur chacun des objets : les 3 points permettent de définir un trièdre sur chaque objet, la classe « `Geo_deplacement` » dispose d'un constructeur calculant la transformation qui applique les trièdres l'un sur l'autre.

Fonction de service annexe :

La fonction

```
int Stl_Maillage::Stl_SubsetLoad(int numObj,int nbSteps,
                                 MemPool*  pPoolOut,BOOL init = TRUE );
```

permet d'extraire un sous ensemble régulier de triangles d'un objet particulier défini par numObj . Les triangles sont extraits le long d'une « ligne de front » progressant régulièrement. Le paramètre nStep définit à la fois le pas des fonts desquels les triangles sont extraits (on ne prélève des triangles qu'un front sur nStep fronts), et pour un front donné, le pas des triangles extraits. En conséquence, si nStep est multiplié par le nombre n, on extrait environ n^2 fois moins de triangles. Les triangles extraits sont rangés dans un « MemPool » sous forme de « codedPointer » . Cette fonction permet d'améliorer les performances : il n'est pas nécessaire d'utiliser tous les point de la triangulation pour la fonction « best_fit » (en particulier lors des premières itérations) .

Acceptation ou refus des points projetés:

La projection orthogonale d'un point d'un maillage sur l'autre maillage présente plusieurs difficultés :

- Un maillage n'est en rien une surface G1 continue, la notion même de projection orthogonale n'est pas clairement définissable.
- Par définition du problème posé, seule une zone du premier maillage peut être mise en correspondance avec le deuxième maillage

Pour résoudre le premier point, la notion de projection orthogonale a été remplacée par la notion de « point le plus proche » qui a toujours une solution, et élimine bon nombre de multiples solutions de la projection orthogonale. Mais cela ne simplifie pas, bien au contraire, la seconde difficulté. L'heuristique suivante a été adoptée :

- La « normale » au premier maillage en un point M du maillage 1 doit être dirigée dans le même sens que la « normale » au deuxième maillage au point M'' (produit scalaire des 2 normales positif)
- L'angle entre la direction MM'' et la «normale » au point M'' doit être inférieur à une valeur donnée (angle non nul lorsque M'', point le plus proche de M se situe sur un côté ou un sommet de triangle)
- M'' ne doit pas se situer sur un bord libre du maillage (tentative de projection d'un point M en dehors de la zone commune) .

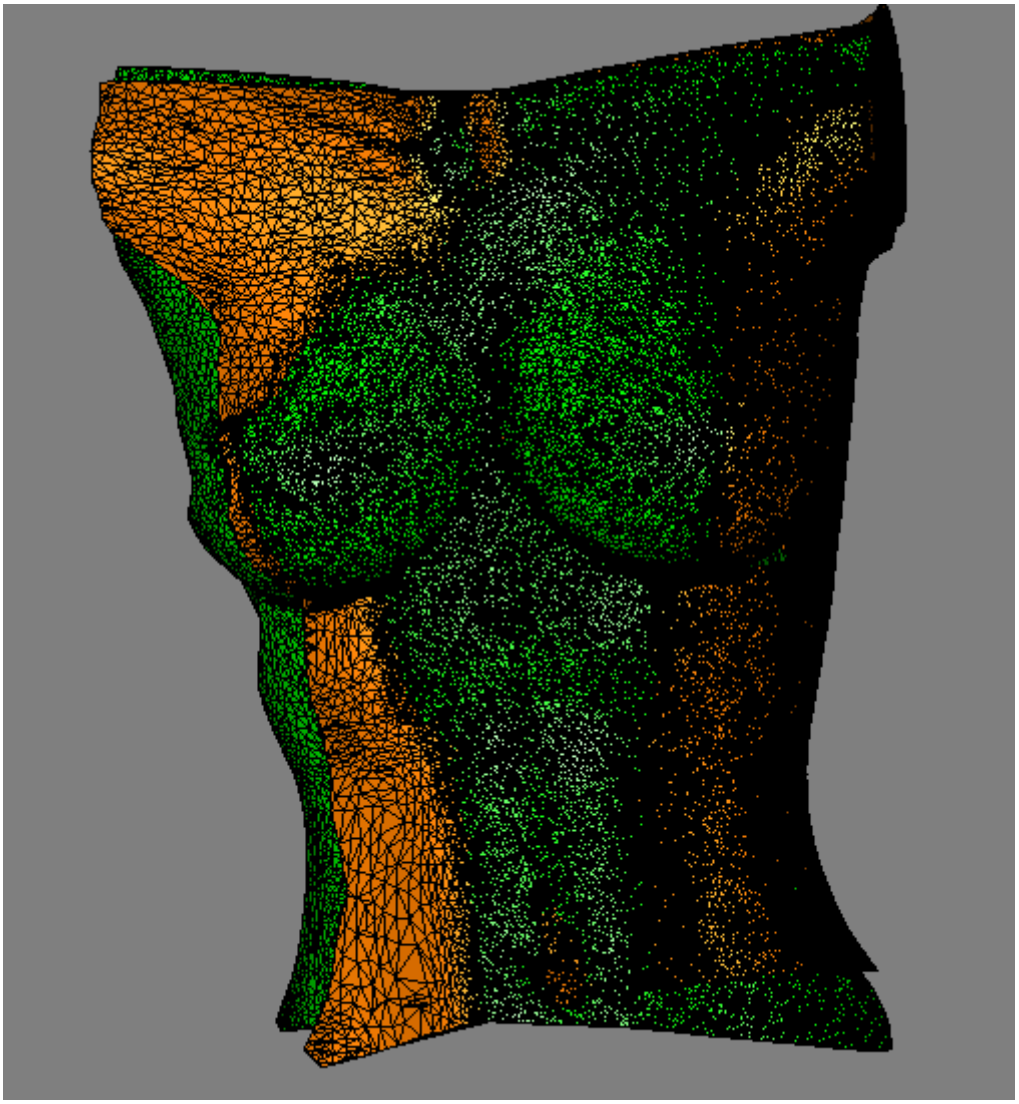
Lorsqu'une de ces conditions n'est pas réalisée, le point ne participe pas au calcul. Lors des itérations finales, il pourrait être opportun d'introduire un critère de distance maximale des points M et M'' (par exemple alfa fois l'erreur quadratique moyenne).

Quelques résultats :

Les tests ont été effectués sur 5 fichiers numérotés de 1 à 5 représentant un buste numérisé sous différents points de vue. Les écarts initiaux entre les repères de 2 fichiers de numéro consécutifs est de l'ordre de 0.1 à 0.2 radians.

On a testé la fonction Best_Fit sur les couples de fichiers (1,2), (2,3), (3,4) et (4,5) directement dans leur version « non alignée » sans aucune intervention manuelle préparatoire. Dans tous les cas on a obtenu un résultat quasi stable après 4 ou 5 itérations, l'erreur quadratique moyenne passant d'une valeur initiale de 10 à 20 selon les cas à une valeur toujours inférieure à1.

Ci-après le résultat du couple (1,2) après 5 itérations.



https://meshprocess.com