Matching de Formes

Julien ALTIERI, Benoit SEGUIN, Frédéric WILHELM

Ecole Polytechnique, X2008

Abstract. Le *matching de formes* est un problème récurrent de géométrie algorithmique. Il s'agit d'une part d'évaluer la *proximité géométrique* de deux maillages indépendamment de ces maillages, et d'autre part de trouver une *transformation rigide* qui envoie l'un sur l'autre, de manière optimale. Le défi consiste donc à extraire des propriétés géométriques *intrinsèques* de l'objet, ce que nous faisons en exploitant le *tenseur de courbure*. **suite à expliquer pour la transformation rigide**.

1 Introduction, motivation, ...

Nous souhaitions au départ développer un jeu de sculpture, avec des formes à reproduire. Cet objectif nous a naturellement conduit à devoir évaluer un score de proximité, et donc au matching de forme.

2 La méthode employée

Nous voulons pouvoir évaluer la correspondance entre deux maillages, et ce de manière invariante par changement d'échelle. Nous nous sommes alors intéressés à la courbure du maillage, et plus précisément à la distribution des courbures. Nous calculons donc une courbure en chaque point du maillage, puis avec ces courbures, une signature. Les méthodes de calcul de courbure sont variées et nous en avons implémenté deux : la courbure de Gauss [**?**] et le tenseur de courbure par la méthode de Taubin [**?**].

3 Courbure de Gauss

La première manière et la plus simple de calculer la courbure en un point est la courbure de Gauss. En version discrète, elle s'exprime par la formule suivante :

$$\kappa_G(v) = \frac{3(\sum_i \theta_i - 2\pi)}{\sum_i Aire(F_i)}$$

Cette formule permet d'obtenir instantanément le produit des courbures principales. Ainsi, en effectuant un passage sur chacun des points on obtient une distribution de la courbure de Gauss du maillage.

Comme l'on essaye de traiter le problème en étant indépendant par rapport à l'échelle, on impose une certaine moyenne à la distribution des courbures. On divise donc toutes les valeurs par la moyenne désirée, puis grâce à un *arctan* on peut transposer l'intervalle non borné des courbures dans un intervalle $[-1; 1]$.

4 Taubin

4.1 Évaluer le tenseur de courbure

Une seconde méthode consiste à utiliser la méthode de Taubin [**?**] pour évaluer un tenseur de courbure sur notre maillage. Ce tenseur représente l'application bilinéaire en un point du maillage, qui à un vecteur direction (dans le plan tangent à la surface) associe la valeur de la courbure suivant cette direction. C'est un donc tenseur 2x2 que l'on va diagonaliser. Sa base de diagonalisation est constituée des deux directions principales de courbure (orthogonales dans cette approximation). Les valeurs propres associées sont les courbures selon ces directions et servent à comparer deux maillages.

4.2 Calcul de la signature

Dans le cas d'estimation de courbure précédent (courbure de Gauss), à chaque point correspondait une unique courbure, donc un unique nombre. L'établissement d'une signature consistait donc à recenser les différentes courbures. Dans le cas présent, chaque point est caractérisé par deux valeurs propres du tenseur de courbure. Il s'agira donc d'effectuer un recensement en deux dimensions.

On implémente donc la signature par une matrice de valeurs réelles à deux dimensions $S_{i,j}$, de taille donnée selon la résolution voulue, par exemple 512×512. Pour chaque vertex v ayant comme courbures $c1$ et $c2$, on va incrémenter $S_{i,j}$ où i et j sont fonction de v (on incrémente également $S_{j,i}$ car les deux valeurs propres ne sont pas ordonnées). Chaque point "vote" donc pour un couple de courbure :

$$S_{i,j} = S_{j,i} = \#\{v \in V | f(c_1(v)) = i \text{ et } f(c_2(v)) = j\}$$

où f est une fonction à définir.

Pour déterminer f, il est nécessaire de s'intéresser aux problèmes d'échelle et d'échantillonnage : en effet, même si deux maillages que l'on souhaite comparer ont intuitivement et visuellement "la même forme", rien ne dit qu'ils ont la même échelle, ni le même échantillonnage (nombre de vertex). Pire encore, il se peut que pour deux formes similaires, les densités d'échantillonnage varient au sein de la même forme, et perturbent le vote.

Pour l'échantillonnage, le problème se résout en pondérant les votes des points par leurs poids, où l'on définit le poids w d'un sommet par la somme des surfaces des faces avoisinantes :

$$\forall v \in V, w(v) = \sum_i A(f_i),$$

Où les f_i sont les faces voisines de v, et $A(f_i)$ leurs aires.

Quand à l'indépendance vis-à-vis de l'échelle, nous avons choisi d'effectuer, pour chaque maillage, une homothétie sur l'ensemble des courbures de telle sorte que la moyenne soit la même. Ainsi, la carte des courbures n'est pas modifiée si on agrandit la forme.

Formellement, l'algorithme de construction de la signature est le suivant :

```
MC : moyenne des courbures
Pour chaque vertex v
   Soient c1(v), c2(v) les valeurs propres de courbure
   Soient i = floor(c1(v) / MC), j = floor(c2(v) / MC)
   Faire signature(i,j) += w(v)
```

On termine en appliquant un flou gaussien sur la signature, qui est au départ très bruitée.

5 Comparer deux courbures

Les méthodes présentées nous fournissent un *descripteur de courbure*, mais comment les comparer ? Notre difficulté majeure réside dans le choix de la distance à utiliser : si nos descripteurs sont remis à l'échelle en alignant les moyennes de chaque distribution, la simple distance euclidienne entre les deux vecteurs à comparer nous donne des résultats vraiment décevants. Par exemple, essayant de corréler `tanglecube.off` avec `tanglecube_fin.off` (censés se ressembler très fortement) et `tanglecube.off` avec `skull.off` (n'ayant vraiment rien à voir), les deux distances n'ont qu'un rapport de 5 environ. Conceptuellement, deux pics "éloignés" dans la distribution sont aussi mal corrélés que deux pics "proches".

6 Organisation du code

Nous choisissons d'implémenter deux versions différentes de la courbure, nous utilisons donc une classe abstraite `CourbureEstimator` qui comporte essentiellement une méthode `double compareTo(CourbureEstimator c)` renvoyant la distance entre les deux maillages, et une méthode `void computeSignature()` qui évalue la courbure suivant la méthode utilisée. Deux classes l'implémentent : `GaussianCourbureEstimator`

pour la méthode de Gauss, et `Taubin` pour le tenseur de courbure. Le programme principal se situe dans la classe `Matching`

7 Conclusion