TP2 - Géométrie des surfaces

Alexandra Bac

Modélisation géométrique
Polytech Marseille - IRM 4A

Ce TP fait suite au premier TP sur les maillages. Suite au cours, vous en savez maintenant plus sur la géométrie des surfaces : normales, courbures ...

Attention, cette fois, comme les calculs impliquent la résolutions de problèmes d'algèbre linéaire, il sera nécessaire d'utiliser également la bibliothèque **Eigen**. Il faudra donc jongler entre les points / vecteurs / matrices de CGAL et ceux d'Eigen ...

1 Calcul des courbures par ajustement de patchs quadratiques

1.1 Introduction

Installation Récupérez le matériel de TP (si nécessaire vous pourrez récupérer les fonctions que vous aviez développées dans le *starting code*).

Présentation de l'approche Durant le TP 1, vous avez implémenté une formule de calcul de la courbure gaussienne en un point d'un maillage. Vous voyez maintenant qu'elle est très "rustique". Nous allons ici implémenter une approche beaucoup plus précise, robuste et complète passant par l'approximation du maillage par une surface lisse. Pour cela, le matériel de TP vous fournit l'ajustement aux moindres carrés d'un patch quadratique en un point (et ses voisins).

La figure 1 illustre l'approche permettant d'estimer le tenseur de courbures au point P :

1. Calculer un repère local $(\vec{u}, \vec{v}, \vec{n_P})$:
   (a) $\vec{n_P}$ estimée à partir du maillage (via les normales des faces voisines, cf. cours)
   (b) $\vec{u}, \vec{v}$ base orthonormale du plan tangent
2. Récupérer les voisins de P (premiers ou deuxièmes voisins) - on obtient un ensemble de points $\{P_i : i \in I\}$ - leur appliquer un changement de base vers $(P, \vec{u}, \vec{v}, \vec{n_P})$.
3. Dans cette base, on va calculer la surface cartésienne quadratique approximant cet ensemble de points aux moindres carrés. Son équation est donc :

$$z = a_0 x^2 + a_1 xy + a_2 y^2 + a_3 x + a_4 y$$

   Le minimum sera calculé par SVD.
4. Une fois la surface calculée, le tenseur de courbure du maillage au point P est estimé par celui de la surface cartésienne.

Afin de vous aider à aller au bout de cette approche, de larges part de code ont déjà été pré-implémentées dans la matériel fourni avec ce TP.

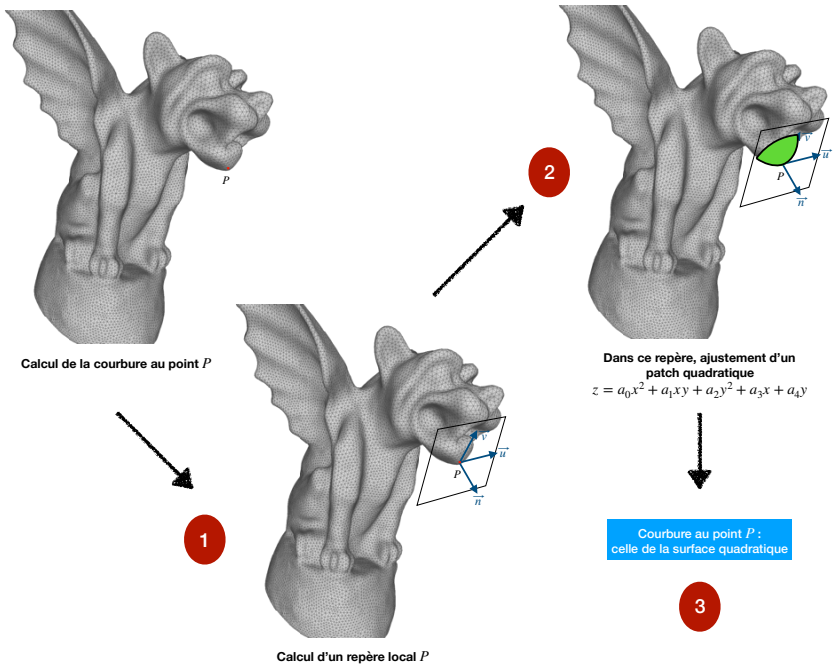
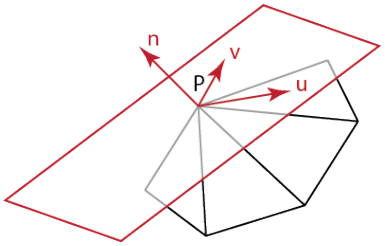


Figure 1: Etapes du calcul

1.2 Description détaillée

- Pour calculer un repère local :
  - On calcule tout d'abord la normale $\vec{n}$ au sommet P (une version basique est implémentée dans la classe)
  - Puis on se place dans le repère local $(P, \vec{u}, \vec{v}, \vec{n})$ où (u, v) est une base orthonormale du plan tangent :



    Le passage de la base canonique à ce repère se fait par une translation $\vec{t}$ et une rotation r (calculées dans la fonction `fit_quad`).

- Dans ce repère, on va alors ajuster une surface d'élévation quadratique passant par P aux points du cercle des premiers ou seconds voisins. L'équation de cette surface est :

$$z = g(x, y) = a_0x^2 + a_1xy + a_2y^2 + a_3x + a_4y$$

  Ayant 5 coefficients, il faut au moins 5 points (plus est mieux) pour pouvoir ajuster une quadrique. Dans `get_two_neighborhood` vous trouverez le code de calcul du 2-voisinage d'un point. Vous l'ajusterez pour vous contenter des premiers voisins quand il y en a assez ... Peut-on appliquer cette méthode sur des maillages "sparse" ?

On note $\{X_i = (x_i, y_i, z_i)\}_{i=1\ldots N\}}$ cet ensemble de points (P et ses voisins). L'erreur au $i$ème point est évaluée comme :

$$\varepsilon_i = (g(x_i, y_i) - z_i)$$

On va donc déterminer la fonction g (dépendant des paramètres $(a_0, \ldots, a_4)$) minimisant (moindres carrés) :

$$J(a_0, \ldots, a_4) = \sum_{i=1}^{N} \varepsilon_i^2$$

Déterminez la matrice A et le vecteur B tels que le vecteur d'erreur s'écrive :

$$\begin{pmatrix} \varepsilon_1 \\ \vdots \\ \vdots \\ \vdots \\ \varepsilon_N \end{pmatrix} = A \cdot \underbrace{\begin{pmatrix} a_0 \\ \vdots \\ a_4 \end{pmatrix}}_{X} - B$$

Le problème d'optimisation aux moindres carrés s'écrit alors :

$$J(a_0, \ldots, a_4) = \|AX - B\|^2$$

Le calcul le plus efficace et stable du minimum de ce problème de moindres carrés linéaires se fait par décomposition en valeurs singulières, ie. SVD (ou transformée QR, c'est équivalent). Vous trouverez cette résolution dans la fonction `fit_quad`. On obtient alors les coefficients $X = (a_0, \ldots, a_4)$ optimaux.

1.3 Squelettes de classes fournis

Les fichiers `courbures.h` et `courbures.cpp` implémentent différentes classes :

- Une classe `MyQuad` codant les surfaces quadratiques (changement de repère du repère global au repère local et équation du patch quadratique)
- Quatre classes codant les maillages avec des propriétés stockant les propriétés nécessaires au TP :
  - Classe `MyMesh` constituant une base pour les autres classes. Avant toute chose, un objet `MyMesh` doit être construit en début de votre code.
  - Classe `MyMesh_Norm` ajoutant aux maillages des propriétés de couleur et normale par sommet.
  - Classe `MyMesh_Courb` ajoutant à un `MyMesh_Norm` des propriétés stockant les courbures et directions principales par sommet.
  - Classe `MyMesh_Quad` ajoutant à un `MyMesh_Norm` une propriété `q` stockant en chaque sommet un `MyQuad`

Dans toute la suite, n'oubliez pas qu'il existe maintenant un repère global et en tout point P, un repère local dans lequel la normale (évaluée par moyenne des normales des faces voisines) au point P est envoyée sur l'axe Oz et le point P envoyé sur l'origine 0.

2 A vous ...

Le code d'ajustement d'un patch quadratique est fourni. Tout ce qu'il vous reste à faire, c'est de faire les calculs pour en déduire les bonnes valeurs de courbures.

1. Calcul de la courbure locale du patch en utilisant le cours sur la géométrie des surfaces (comme on travaille sur le patch quadratique ajusté en un point P, ce point a été envoyé en 0 dans le repère local et la normale précédemment estimée a été envoyée sur Oz).

   Pour rappel, l'équation cartésienne du patch quadratique local est :

   $$z = a_0x^2 + a_1xy + a_2y^2 + a_3x + a_4y$$

   (a) Ecrire l'équation paramétrique de cette surface, donc une fonction $f : \mathbb{R}^2 \to \mathbb{R}^3$.

   (b) Calculer les dérivées partielles de f et en déduire les deux vecteurs formant la base du plan tangent en 0.

   (c) En déduire l'expression de la normale en 0, puis de la normale unitaire.

   (d) Comme on avait ajusté l'axe 0z avec une première estimation de la normale (calculée à partir des faces voisines), comment doivent être les coefficients a_3 et a_4.

   (e) Calculer la première forme fondamentale au point 0.

   (f) Calculer les dérivées secondes et en déduire la deuxième forme fondamentale au point 0.

   (g) Que vaut le tenseur de courbure ?

   (h) Complétez les fonctions `quad_first_fond_0`, `quad_sec_fond_0` calculant respectivement la première et deuxième formes fondamentales de la surface quadratique en 0. Que calcule la fonction `courb_tens_0` et comment réalise-t-elle ce calcul ?

   (i) Que font les fonctions :

   - `get_principal_0` ? Que renvoie-t-elle (en particulier quelle est la dimension des vecteurs renvoyés) ?
   - `MyMesh_Quad::princip_curv` ? Que calcule-t-elle ? Comment ? Où stocke-t-elle ses résultats ?

2. On va maintenant tester les résultats obtenus :

   (a) Lancer le code ainsi complété pour obtenir une carte de couleur du maillage par la courbure Gaussienne. Le résultat est-il très différent des résultats obtenus au TP 1 ?

   (b) Revenir au cas de la sphère et analyser les résultats obtenus.

   (c) On veut une comparaison plus précise. Comment calculer une carte de couleur de l'écart entre la courbure gaussienne calculée au premier TP et celle calculée ici ? Les valeurs et la distribution des courbures sont différentes, donc est-ce qu'il suffit de faire la différence ? Proposez une solution qui vous paraît pertinente.

   (Opt) Implémentez cette solution puis analysez les résultats.

   (d) On s'intéresse ensuite au "type" des points, donné par les signes respectifs des courbures κ_1 et κ_2 (convexe, concave ...). Etablissez une carte de couleur (une couleur par type) et colorez le maillage par ces types.

   (e) **(Opt)** Exporter les maillages des patchs quadratiques aux 10 points de courbure la plus importante (5 points de courbure minimale / 5 points de courbure maximale). Comment jugez-vous la qualité de l'ajustement de la surface aux points ?

   (f) **(Opt)** Calculer une carte de courbure moyenne, la comparer à la carte de courbure gaussienne.

   (g) **(Opt)** Proposez un algorithme pour, partant d'un point de courbure importante, calculer une ligne saillante partant de ce point. Testez votre algorithme et présentez des résultats.

(h) **(Opt)** Dans le processus, on dispose de deux normales : la normale estimée sur le maillage (stockée dans la propriété `normal` de `MyMesh_Norm`), et la normale au patch quadratique (voire ci-dessus). Comment devraient être ces normales ? On souhaite les comparer visuellement : définir une mesure de l'écart entre ces deux normales et calculer une carte de couleur de cet écart. Analyser les résultats.

(i) **(Opt)** On souhaite enfin tester la robustesse de la valeur de courbure.

i. Calculer une carte de couleur pour les courbures du maillage, optenues avec les formules du TP1 / TP2 respectivement.

ii. En utilisant le filtre de simplification "Simplification: Quadric edge collapse decimation" et en choisissant le "target number of faces", vous pouvez simplifier votre maillage (choisissez bien l'option "preserve boundary").

A. Construire des modèles simplifiés à 20 et 50% respectivement. Calculer les courbures obtenues par les méthodes du TP1 et TP2 respectivement et commencez par comparer les valeurs et cartes de courbures.

B. Comment évaluer plus précisément la stabilité respective de ces courbures vis à vis de la simplification ?

C. Ecrire une fonction prenant en argument les coordonnées d'un point et un maillage et calculant le sommet du maillage le proche du point.

D. Quelle est la complexité de cette recherche ? En fonction du cours sur les maillages, comment pourriez-vous l'accélérer ?

E. Utiliser cette fonction pour comparer les valeurs de courbures entre maillage initial et maillage simplifié.

F. Essayez d'analyser les résultats.

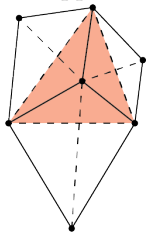
3 Une petite chaîne de traitement de maillages

Le but de cette partie est de pouvoir optimiser un maillage dense. Récupérer le maillage de gargouille fourni sur la page du cours. On veut implémenter un algorithme basique de simplification de maillage dont l'idée est de supprimer des triangles dans les zones assez planes.

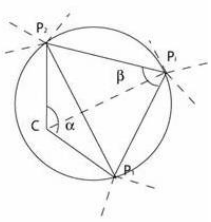
1. Comment pouvez-vous reprérer ces zônes ?

2. Si on souhaite supprimer un sommet du maillage (car sommet sur une zone assez plane), proposez un algorithme simple pour remailler localement.

3. Le remaillage peut produire des triangles étirés. Pour essayer de les éliminer, on peut tenter de permuter les arêtes joignant deux faces (généralement appelé "flip") :



La condition pour décider d'échanger une arête est inspirée par la condition de Delaunay (2D) :



L'arête est échangée si $\alpha + \beta > \pi$.

(a) Ecrire la fonction :

```
bool flip_edge(Mesh::Halfedge_index &he, //
                                        Mesh &m)
```

Testant si la condition est vérifiée pour l'arête attachée à la demi-arête `he`, réalisant l'échange si nécessaire et renvoyant un booléen déterminant si un échange a été réalisé et `he` la référence sur le nouvelle arête.

(b) Utiliser cette fonction localement pour améliorer le maillage produit.

4. Appliquez votre algorithme au maillage et proposez un paramètre utilisateur pour contrôler le taux de simplification (et la qualité de cette dernière).