TP4 - Géométrie des surfaces

Alexandra Bac

Modélisation géométrique
Polytech Marseille - IRM 4A

Ce TP fait suite au premier TP sur les maillages, donc retour à CGAL. 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 ...

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

Présentation du TP 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.

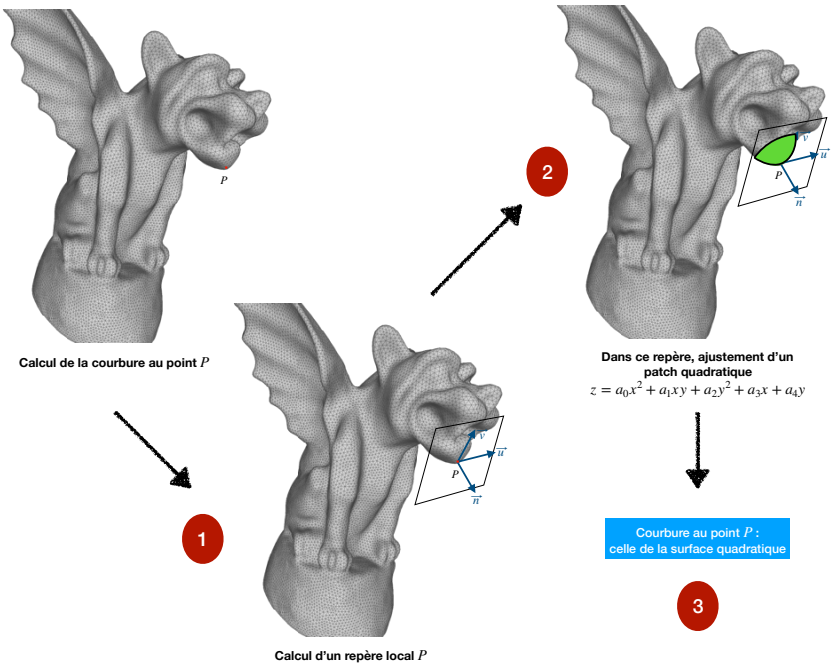


Figure 1: Etapes du calcul

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 :

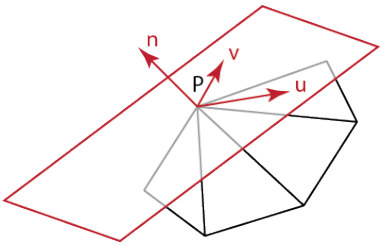
$$z = a_0 x^2 + a_1 xy + a_2 y^2 + a_3 x + a_4 y$$

   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.

1 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 :

$$z = g(x, y) = a_0 x^2 + a_1 xy + a_2 y^2 + a_3 x + a_4 y$$

  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 $\{X_i = (x_i, y_i, z_i)\}_{i=1...N}\}$ 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.

2 Squelettes de classes fournis

Les fichiers `#courburesh` et `courbures.cpp` implémentent différentes classes :

- Une classe `MyQuad` codant les surfaces quadratiques
- 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 stockant une surface quadratique par sommet.

3 A vous ...

1. Il faut, avant toute chose, estimer des normales aux sommets du maillage. Complétez la fonction `compute_normal`, estimant la normale au sommet P comme la moyenne des normales des faces voisines. Puis complétez la fonction `MyMesh_Norm::normales_locales` affectant ces normales à la propriété correspondante.

2. Complétez la fonction `MyMesh_Quad::fit_quad` (vous avez sous la main la matrice de changement de base vers la base locale au sommet P, il suffit de la multiplier aux points pour effectuer le changement de base) pour réaliser l'ajustement de la surface quadratique (et renvoyer une quadrique).

3. 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 fait la fonction `quad_sec_fond_0` ? (cf. cours)

4. Que font les fonctions :

   - `get_principal_0`
   - `Courbures_princip::compute_principals`
   - `Courbures_princip::compute_curvatures`

   Interprétez leurs codes.

5. Enfin assemblez correctement tous ces ingrédients dans `code_courb.cpp` pour réaliser le calcul et exploiter les résultats :

   (a) Construisez un objet `MyMesh_Norm` et assurez-vous que les normales sont bien initialisées.

   (b) Créez des objets `MyMesh_Courb` et `MyMesh_Quad` puis :

      i. Calculez la surface quadratique en chaque point

      ii. En utilisant `MyMesh_Courb::set_courbs` et en lui passant la bonne fonction en argument, calculer et stocker les courbures principales en chaque point

      iii. Quelle fonction permet de récupérer K et H

      iv. Créer une carte de courbures à partir de K

      v. En utilisant la méthodes d'export de maillages de `MyMesh_Norm` et `MyMesh_Courb`, vous pourrez exporter deux maillages : le maillage coloré par cette carte de courbures et le champ des vecteurs de directions principales.

   (c) En utilisant la fonction `local_mesh`, vous pourrez enfin exporter aux points où vous le souhaitez un maillage de la surface quadratique approximante : sélectionnez bien les points où il est intéressant de visualiser cette surface. Petit indice quand `qp` est un MyQuad :

   ```
   Mesh &mi(qp->local_mesh(.1, 8)) ;
   CGAL::IO::write_polygon_mesh("test_quad.off", mi) ;
   ```

   J'ai laissé quelques indices dans le fichier ...

   (d) Enfin, choisissez une couleur par type de sommet (nous avons vu en cours qu'en fonction des signes de κ_1 et κ_2 les sommets peuvent être classifiés en "convexe", "concave", "selle" ... vous colorerez chaque type de sommet d'une couleur).