TP6 – *Shape-from-shading*

Équation eikonale

Soit $\mathbf{Q}$ un point de la surface d'une scène 3D opaque, d'image $\mathbf{q} = [x, y]^\top$. La *loi de Lambert* exprime le niveau de gris $I(x, y)$ de $\mathbf{q}$ sous une forme très simple, qui est en fait une relation de proportionnalité :

$$I(x, y) = \rho(\mathbf{Q})\ \mathbf{n}(\mathbf{Q}) \cdot \mathbf{s}(\mathbf{Q}) \tag{1}$$

où $\rho(\mathbf{Q}) \in \mathbb{R}^+$ désigne l'albédo de $\mathbf{Q}$, $\mathbf{n}(\mathbf{Q})$ désigne la normale sortante unitaire en $\mathbf{Q}$, et $\mathbf{s}(\mathbf{Q})$ caractérise l'éclairage en $\mathbf{Q}$. Un éclairage uniforme *frontal* d'intensité ϕ_0 a pour coordonnées, dans le repère caméra :

$$\mathbf{s}(\mathbf{Q}) = [0, 0, -\Phi_0]^\top \tag{2}$$

En supposant la projection orthogonale, c'est-à-dire en négligeant la perspective, le point $\mathbf{Q}$ a pour coordonnées $\mathbf{Q} = [x, y, z(x, y)]^\top$ dans le repère caméra. La normale $\mathbf{n}(\mathbf{Q})$ s'exprime donc sous la forme suivante (voir cours) :

$$\mathbf{n}(\mathbf{Q}) = \frac{1}{\sqrt{|\nabla z(x, y)|^2 + 1}} \begin{bmatrix} \nabla z(x, y) \\ -1 \end{bmatrix} \tag{3}$$

En supposant de plus $\rho(\mathbf{Q}) = \rho_0$ (albédo uniforme), nous tirons des équations (1), (2) et (3) :

$$I(x, y) = \frac{\rho_0\, \Phi_0}{\sqrt{|\nabla z(x, y)|^2 + 1}} \tag{4}$$

Sachant que $\rho_0\, \Phi_0 = \max\{I(x, y)\} = I_{\max}$, nous tirons enfin de (4) :

$$|\nabla z(x, y)| = \sqrt{\left[\frac{I_{\max}}{I(x, y)}\right]^2 - 1} \tag{5}$$

Cette équation, appelée *équation eikonale*, constitue le modèle le plus simple de *shape-from-shading*.

Résolution de l'équation eikonale

L'équation eikonale nous informe sur la norme de ∇z en chaque point $\mathbf{q} = [x, y]^\top$, mais pas sur sa direction ni son orientation. En d'autres termes, parmi les deux angles qui caractérisent la normale, seul son angle θ avec l'axe Oz, appelé *colatitude*, est connu : en effet, nous ne disposons d'aucune information sur l'autre angle ϕ, appelé *azimut*. L'expression (3) nous montre que ϕ est également l'angle polaire de $\nabla z(x, y)$, relativement au repère image Oxy.

Une façon de rendre bien posée la résolution de l'équation eikonale (5) consiste à imposer deux contraintes à l'inconnue $z(x, y)$. La première contrainte est la *condition au bord* $z(x, y) = 0$ en chaque point du bord de l'image, qui constitue donc une *courbe de niveau*. Le gradient de $z(x, y)$ en ces points est orthogonal au bord de l'image, ce qui permet *presque* de calculer la normale $\mathbf{n}(\mathbf{Q})$. En effet, une ambiguïté résiduelle porte sur le sens de $\nabla z(x, y)$. La deuxième contrainte consiste à orienter $\nabla z(x, y)$ vers le centre de l'image. Il est donc possible de calculer une nouvelle courbe de niveau $z(x, y) = c$, où $c > 0$ est un réel « petit ». En répétant l'opération de proche en proche, on peut alors calculer la solution $z(x, y)$ sur l'ensemble de l'image.

Comme l'hypothèse $z(x, y) = 0$ ne correspond généralement pas à la réalité (la scène 3D n'a pas toujours un bord plan !), il ne faut pas s'attendre à ce que cette solution, appelée *solution de viscosité maximale*, soit réaliste, mais il s'agit d'une solution exacte de l'équation eikonale, qui reproduit fidèlement l'image initiale sous un éclairage frontal, à ceci près que la solution est différentiable *presque partout*. Cela se traduit par le fait que le relief reconstruit comporte généralement des arêtes, qui sont facilement visibles lors du rééclairage.

Vous allez programmer deux méthodes d'approximation discrète de la solution de viscosité maximale. La première de ces méthodes est itérative (cf. exercice 1), tandis que la deuxième, appelée *fast marching*, calcule directement la solution de proche en proche, par le biais de l'évolution d'un « front d'onde » (cf. exercice 2). Cette dernière méthode a été proposée par Tsitsiklis en 1995, puis reprise par Sethian en 1996.

Exercice 1 : résolution itérative de l'équation eikonale

Pour trouver une approximation discrète de la solution de viscosité maximale de l'équation eikonale (5), on peut utiliser les approximations suivantes des deux dérivées partielles de z par *différences finies centrées* (nous supposons ici que $x \equiv i$ et $y \equiv j$, ce qui n'est pas le cas en Matlab) :

$$\left(\frac{\partial z}{\partial x}\right)_{i,j} \approx \frac{z_{i+1,j} - z_{i-1,j}}{2} \quad ; \quad \left(\frac{\partial z}{\partial y}\right)_{i,j} \approx \frac{z_{i,j+1} - z_{i,j-1}}{2} \tag{6}$$

Si nous désignons par $f_{i,j}$ le membre droit de (5) calculé au pixel (i,j), nous pouvons trouver une approximation discrète de la solution de viscosité maximale de l'équation (5) à l'aide du schéma numérique suivant :

$$z_{i,j}^{(k+1)} = z_{i,j}^{(k)} + a\left\{f_{i,j} - \sqrt{\left(\frac{z_{i+1,j}^{(k)} - z_{i-1,j}^{(k)}}{2}\right)^2 + \left(\frac{z_{i,j+1}^{(k)} - z_{i,j-1}^{(k)}}{2}\right)^2}\right\} \tag{7}$$

où k indique le pas de l'itération et $a > 0$ est un paramètre. En effet, si ce schéma converge vers z^*, alors :

$$z_{i,j}^{*} = z_{i,j}^{*} + a\left\{f_{i,j} - \sqrt{\left(\frac{z_{i+1,j}^{*} - z_{i-1,j}^{*}}{2}\right)^2 + \left(\frac{z_{i,j+1}^{*} - z_{i,j-1}^{*}}{2}\right)^2}\right\} \tag{8}$$

En simplifiant (9) et en utilisant (6), il s'avère que z^* est bien une solution de l'équation eikonale, puisque :

$$\sqrt{\left(\frac{z_{i+1,j}^{*} - z_{i-1,j}^{*}}{2}\right)^2 + \left(\frac{z_{i,j+1}^{*} - z_{i,j-1}^{*}}{2}\right)^2} = f_{i,j} \tag{9}$$

Malheureusement, le schéma (7) est numériquement instable. On lui préfère un schéma de type « Lax-Friedrichs », où $z_{i,j}^{(k)}$ doit être remplacé par la moyenne de la profondeur de ses quatre plus proches voisins, dont on montre qu'il est stable dès lors que $a \leq 0,5$ (en pratique, on fixe a à $0,5$) :

$$z_{i,j}^{(k+1)} = \frac{z_{i-1,j}^{(k)} + z_{i+1,j}^{(k)} + z_{i,j-1}^{(k)} + z_{i,j+1}^{(k)}}{4} + a\left\{f_{i,j} - \sqrt{\left(\frac{z_{i+1,j}^{(k)} - z_{i-1,j}^{(k)}}{2}\right)^2 + \left(\frac{z_{i,j+1}^{(k)} - z_{i,j-1}^{(k)}}{2}\right)^2}\right\} \tag{10}$$

Écrivez la fonction `lax_friedrichs`, appelée par le script `exercice_1`, permettant de résoudre l'équation eikonale (5) par le schéma itératif (10), **qui doit être appliqué seulement aux pixels du masque**. En guise d'initialisation de la profondeur, vous pourrez choisir $\{z_{i,j}^{(0)}\}$ uniformément égale à 0. Comme cela a déjà été dit, la reconstruction 3D obtenue n'est pas une approximation réaliste du relief, car les hypothèses sur l'albédo et sur l'éclairage ne sont pas réalistes. Néanmoins, l'image simulée par rééclairage frontal reproduit fidèlement l'image d'origine, à part le long de quelques arêtes qui sont inhérentes aux solutions de viscosité.

Une fois la fonction `lax_friedrichs` mise au point, vous pourrez tester d'autres valeurs du paramètre a. Vous pourrez également constater l'instabilité du schéma numérique (7), y compris lorsque $a = 0,5$.

Exercice 2 : résolution directe de l'équation eikonale

La résolution de l'équation eikonale par l'algorithme du *fast marching* revient à simuler l'évolution d'un « front d'onde », à une vitesse égale au second membre de l'équation (5). Au démarrage de l'évolution, le front d'onde doit être connu. Vous allez bien sûr utiliser la condition au bord $z(x,y) = 0$ sur le bord de l'image.

Les pixels sont répartis en trois classes, qui forment une partition de l'image :
- la classe *calculés* contient les pixels où la valeur de la profondeur z a déjà été calculée ;
- la classe *proches* contient les pixels dont au moins un des 8 plus proches voisins a déjà été calculé ;
- la troisième classe contient tous les autres pixels.

Cet algorithme est rapide car il n'est pas itératif : chaque pixel est « visité » une seule fois. L'astuce consiste à calculer les valeurs de la profondeur z de manière à former une suite croissante. Le prochain pixel où la profondeur z est calculée doit appartenir à la classe *proches*. Une fois le calcul effectué, ce pixel passera dans la classe *calculés*. Pour déterminer le prochain pixel *proche* qui passera dans la classe *calculés*, il faut calculer, pour chaque pixel *proche* $\mathbf{p}_1$:

$$\min_{\mathbf{p}_2 \in \mathcal{V}(\mathbf{p}_1)} \{ z(\mathbf{p}_2) + \delta z(\mathbf{p}_1, \mathbf{p}_2) \} \tag{11}$$

où $\mathcal{V}(\mathbf{p}_1)$ contient les pixels voisins de $\mathbf{p}_1$ appartenant à la classe *calculés*, et où la variation de profondeur $\delta z(\mathbf{p}_1, \mathbf{p}_2)$ est égale à la distance entre $\mathbf{p}_1$ et $\mathbf{p}_2$ (qui est égale à 1 ou $\sqrt{2}$, si l'unité de longueur est le pixel) multipliée par la moyenne du gradient de profondeur en $\mathbf{p}_1$ et en $\mathbf{p}_2$:

$$\delta z(\mathbf{p}_1, \mathbf{p}_2) = \|\mathbf{p}_2 - \mathbf{p}_1\| \, \frac{|\nabla z(\mathbf{p}_1)| + |\nabla z(\mathbf{p}_2)|}{2} \tag{12}$$

Écrivez les fonctions `delta_z` et `fast_marching`, appelées par le script `exercice_2`, permettant d'approcher la solution de viscosité maximale de l'équation eikonale (5). Il est conseillé de manipuler les pixels par leurs indices absolus, et d'utiliser deux vecteurs lignes de booléens comportant autant de colonnes qu'il y a de pixels dans l'image, pour tenir à jour les listes de pixels appartenant aux classes *calculés* et *proches*.

Remarques

- Afin d'éviter les divisions par 0 lors de l'estimation de $|\nabla z|$ par l'équation (5), estimation qui est nécessaire pour calculer l'expression (12) de δz, le dénominateur $I(x, y)$ doit être remplacé par $\max\{I(x, y), \epsilon\}$.

- Il est conseillé de prendre connaissance de la fonction `voisinage`, qui vous est fournie, afin de bien comprendre comment les paires de pixels voisins sont organisées dans la matrice `voisins`.