Algorithme de Dijkstra et Fast Marching

Tom Février, Alexandre Libourel, Mattéo Oziol

1 Introduction

L'algorithme de Dijkstra est un algorithme de recherche de chemin dans une image qui permet de résoudre le problème du plus court chemin entre deux points sur une image. La valeur prise par chacun des pixels représente une topographie de terrain, ou une viscosité. L'algorithme a été conçu par le scientifique informatique Edsger W. Dijkstra en 1956 et publié trois ans plus tard. L'algorithme existe sous plusieurs variantes ; la variante originale de Dijkstra trouvait le plus court chemin entre deux noeuds d'un graphe. Ici on considère notre image comme notre graphe dont chaque pixel représente les différents noeuds du graphe. La notion de voisinage est remplacée par celle de pixel adjacent : Deux pixels sont voisins ssi ils sont adjacents.

L'algorithme fonctionne en maintenant un ensemble de pixels pour lesquels le plus court chemin depuis le pixel source a déjà été déterminé. Il sélectionne répétitivement le pixel de cet ensemble avec la distance connue la plus petite par rapport à la source, calcule la distance à travers ce noeud jusqu'à chacun de ses voisins non visités et met à jour la distance du voisin si elle est plus petite. Ce processus continue jusqu'à ce qu'il n'y ait plus de pixels dans l'ensemble, moment auquel l'algorithme a déterminé le plus court chemin vers tous les pixels de l'image.

L'algorithme de Dijkstra est un exemple d'algorithme glouton, car il prend la décision la plus avantageuse localement à chaque étape dans l'espoir de trouver une solution globale optimale. Il est utilisé pour trouver les plus courts chemins entre les pixels d'une image, qui peuvent par exemple représenter des réseaux de routes. Pour un pixel source donné dans l'image, l'algorithme trouve le plus court chemin entre ce pixel et tous les autres. Il peut également être utilisé pour trouver les plus courts chemins entre un seul pixel et un seul pixel de destination en arrêtant l'algorithme une fois le plus court chemin vers le pixel de destination déterminé.

2 Pseudo Code de l'algorithme de Dijkstra

Voici le pseudo Code de l'algorithme de Dijkstra. Où :

— W **:** L'image d'entrée. Il s'agit d'une image où chaque pixel représente la viscosité du terrain. Plus la valeur du pixel est élevée, plus la viscosité est importante.

— x_0 **:** Le point initial. Il s'agit du point de départ de l'algorithme. Toutes les distances calculées seront des distance d'un point x_1 à x_0.

— D **:** La matrice des distances au point x_0. Il s'agit de la sortie de l'algorithme.

— S **:** La matrice des pixels visités. Chaque cellule de la matrice prend une valeur dans $\{-1, 0, 1\}$. 0 si le pixel correspondant n'a pas été visité, 1 si celui-ci a été visité et -1 si celui-ci a été visité et que le plus court chemin de ce pixel à celui de départ a été trouvé.

— $distance(i, j, D, W)$ **:** Une fonction de calcul de distance entre deux pixels d'indice i et j, sachant la distance de x_0 à x_i grâce à la carte des distances D et la matrice de viscosité W.

Remarque :
En effet, plusieurs fonction de distance peuvent être utilisées. La fonction la plus naïve revient à ajouter la distance la valeur de $W[j]$ à la distance de j. Cependant une telle fonction induit une distance $L1$ entre les points. Pour plus de réalisme, on peut introduire des fonctions de distances plus fines détaillées dans le notebook.

Algorithm 1 Algorithme de Dijkstra

Entrée : Matrice pondérée W, Point initial x_0
$S \leftarrow$ Matrice des points visités (initialisée à 0)
$D \leftarrow$ Matrice des distances au point x_0 (initialisée à ∞)
$I \leftarrow$ Liste des indices pixels voisins non visités (initialisée à $[x_0]$)
$S[x_0] \leftarrow 1$
$D[x_0] \leftarrow 0$
while I n'est pas vide **do**
  Déterminer l'indice j du pixel de D ayant la plus petite valeur parmi la liste d'indice I.
  Extraire i de I
  $S[i] \leftarrow -1$
  for chaque voisin j de i **do**
    if $S[j] \neq -1$ **then**
      $distance\_temp \leftarrow distance(i, j, D, W)$
      if $distance\_temp < D[j]$ **then**
        $D[j] \leftarrow alt$
      end if
    end if
    $S[j] \leftarrow 1$
    Ajouter j à I
  end for
end while
Renvoyer D

L'algorithme de Dijkstra permet de créer une matrice des distances partant du point x_0. Une fois cette carte obtenue, on va pouvoir déterminer le plus court chemin entre x_0 et n'importe quel autre point de l'image à l'aide d'un algorithme d'optimisation.

3 Pseudo code de l'algorithme du plus court chemin.

Algorithm 2 Algorithme du plus court chemin

Entrée : La matrice des distances D (relative à x_0) obtenue par l'algorithme de Dijkstra, un point d'arrivée x_1, un pas τ
$G \leftarrow grad(D)$
$\Gamma \leftarrow [x_1]$ #Initialiser le chemin avec le point d'arrivée
while le dernier élément de Γ est loin de x_0 **do**
  $\gamma_{prec} \leftarrow \Gamma[-1]$
  $g \leftarrow Interpolate(\gamma, G)$ #γ n'a pas forcément des coordonnées entières
  $\gamma_{new} \leftarrow \gamma_{prec} - \tau g$
  Ajouter γ_{new} à Γ
end while
Renvoyer la trajectoire Γ

Toutes les différentes étapes de Dijkstra et de l'algorithme du plus court chemin sont illustrées et expliquées dans le notebook.