INF8801A Applications multimédias

Travail pratique 2

Inpainting par recherche dans une base d'images

Département de génie informatique et génie logiciel
Polytechnique Montréal
Automne 2019

Historique des modifications du document

Version	Description	Auteur
1.0	Version initiale	Olivier Pinon & Thomas Hurtut

1 Description globale

1.1 But

Le but de ce TP est d'implémenter la méthode appelée *Scene Completion* décrite dans l'article suivant : http://graphics.cs.cmu.edu/projects/scene-completion/. Il s'agit d'une méthode permettant de combler des trous dans une image (inpainting), en utilisant le morceau d'image le plus *adapté* trouvé dans une très grande base d'images. Ce TP est divisé en deux parties indépendantes : la première consiste en la recherche d'images similaires dans une base ; la seconde consiste en l'inpainting à proprement dit : la composition des deux images.

1.2 Partie de la méthode fournie

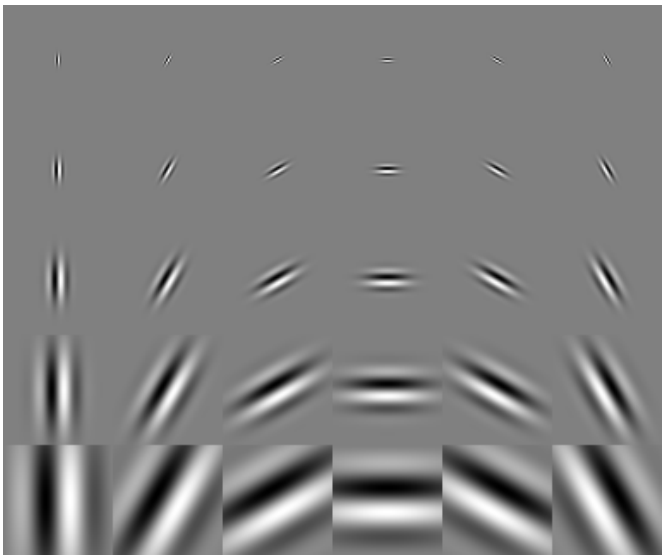
Un canevas vous est fourni, et vous n'aurez à compléter que certaines étapes par du code Matlab. L'algorithme global de *Scene Completion* (y compris les parties que vous n'aurez pas à implémenter) se décompose comme ceci :

- Calcul du descripteur GIST sur la base de données (**question 1**)
- Requête des images les plus proches dans la base
- Translation optimale
- Calcul du découpage optimal avec GraphCut
- Composition avec l'algorithme de Poisson (**question 2**)

2 Travail demandé

2.1 Recherche dans la base de données

Dans cette partie, nous allons étudier le descripteur utilisé sur cette base de données. Il a deux composantes : la première décrit les couleurs de l'image, la seconde décrit les contours de l'image : il s'agit de *GIST*.



$$exp(i * (2 * \pi * \frac{x'}{\lambda} + \psi)) * exp(-\frac{x'^2+\gamma^2*y'^2}{2*\sigma^2})$$

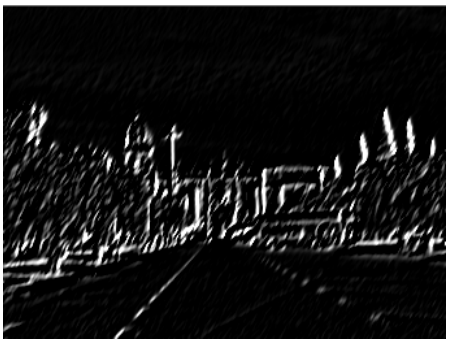
où $x' = x * cos(\theta) + y * sin(\theta)$

et $y' = -x * sin(\theta) + y * cos(\theta)$

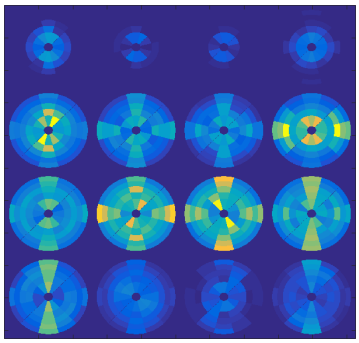
(a) Formule générale d'un filtre de Gabor

FIGURE 1 – Filtres de Gabor (à différentes échelles et orientations)

Image en niveaux de gris



Réponse à un filtre horizontal



Descripteur Gist

FIGURE 2 – Exemples de réponses aux filtres de Gabor. Le graphique à droite représente la réponses de chacune des 4x4 régions de l'image aux filtres. Vous pouvez obtenir ce graphique avec la fonction descGist.display(). Chaque rosace correspond au set de filtres de Gabor (à différentes échelles et orientations).

Question 1 – Descripteur GIST

Pour cette question, vous devez implémenter la fonction *descGist.m* qui prend en entrée une image (à mettre en niveaux de gris) et qui retourne la réponse de l'image à 30 filtres de Gabor (6 angles et 5 échelles, c.f. figure 1). La valeur absolue de la réponse de chaque filtre est moyennée sur les régions de l'image correspondant à une grille de 4x4 (attention à la manière dont vous moyennez ; n'utilisez pas de filtre *Nearest*). Nous obtenons donc un total de 480 valeurs. Ces valeurs peuvent être affichées sous forme de graphique avec la fonction *descGist.display()* qui vous est donnée (c.f. figure 2).

2.2 Composition des deux images

Dans cette partie, une image similaire à l'image de requête vous est donnée. Vous devez assembler les deux images afin de combler au mieux le trou dans l'image requête.

Question 2 – Algorithme d'édition de Poisson

Il s'agit de la dernière étape de l'algorithme. Vous devez implémenter la fonction *PoissonBlending.m* qui prend en entrée 3 arguments :

- L'image à combler
- L'image avec laquelle la combler (même dimensions)
- Un masque représentant les pixels à combler

Vous utiliserez pour cela l'algorithme de Poisson, décrit dans cet article : *Poisson Image Editing [Perez et al. 2003]*. Le but de cette méthode est de coller l'image dans l'espace du Gradient, puis de l'intégrer pour obtenir le résultat du collage ; cette intégration peut être approximée par la méthode de Poisson (en résolvant le problème $\triangle I = \operatorname{div} G$ où G est le gradient de l'image de référence.).
Une implémentation possible est d'utiliser la méthode de Jacobi. En effet, résoudre un Laplacien nul sur un ensemble de pixels revient à itérativement assigner comme valeur à chaque pixel la moyenne de celles de ses voisins à l'itération précédente (figure 3).

FIGURE 3 – Évolution de la diffusion par le Laplacien (Méthode de Poisson)

3 Fichiers fournis

Nom fichiers	**Description**
mainGist.m descGist.m	Code matlab pour le descripteur GIST
mainPoisson.m poissonBlending.m	Code matlab pour le collage par la méthode de Poisson
data/	Images de test

4 Exigences

E1. Question 1 : Compléter la classe `descGist.m`
qui est utilisée par le script `mainGist.m`

E2. Question 2 : Compléter la fonction `poissonBlending.m`
qui est utilisée par le script `mainPoisson.m`