Programmation DM 1 : *Seam Carving*

{ jrobert, fbouchez, apardon } @ens-lyon.fr
http://perso.ens-lyon.fr/florent.bouchez/

12 octobre 2006

Formalités : *ce devoir est à faire en binômes, et à rendre avant le* ***lundi 12 novembre 2006****.*

Vous devez nous envoyer par e-mail un rapport (LaTeX compilé en `ps` *ou* `pdf`*) et une archive* `tar.gz` *contenant tous vos fichiers. Si vous n'êtes pas encore à l'aise avec LaTeX, vous pouvez aussi nous rendre le rapport sur papier. Indiquez bien vos noms dans le rapport et l'archive (si possible, nommez-les* `login1-login2.pdf` *et* `login1-login2.tar.gz`*). Attendez-vous aussi à devoir faire une mini-présentation de votre œuvre les jours suivants.*

> **Avertissement :** dans ce DM, vous êtes un peu lâchés dans la nature par rapport aux TPs. Il est donc normal que ça vous paraisse difficile. C'est aussi pour ça qu'on vous laisse un mois pour ce devoir. Il est important que vous y jetiez un coup d'œil rapidement et que vous le commenciez tôt. Si vous êtes bloqués quelque part, n'hésitez surtout pas à nous en parler nous sommes là pour ça : par e-mail, à la fin d'un TP. Enfin, ce travail est différent des habitudes scolaires que vous avez pu avoir, il est donc normal que vous ayez des connaissances très disparates les uns des autres : n'ayez pas peur et posez-nous des questions, même si elles vous paraissent stupides.

Introduction

Pour faire tenir une image dans un certain espace, il existe plusieurs méthodes simples :

- le redimensionnement : on projette de façon linéaire l'image de départ dans un cadre plus petit.
- le recadrage : on garde une partie de l'image (souvent la plus intéressante selon certains critères) de la dimension désirée. Cela demande souvent une intervention humaine.

Le redimensionnement supprime arbitrairement des informations de l'image quelles que soient leur importance. Le recadrage ne permet pas forcement de garder les informations essentielles (par exemple deux éléments très éloignés dans une image).

Il existe une autre méthode :

Observez bien l'exemple donné : les parties situées entre et à côté des deux tours ont été estimées moins importantes que les tours elles-mêmes et ont donc été rétrécies, tandis que les tours ont gardé leur taille et forme d'origine. Cette nouvelle méthode a été présentée cette année à la conférence SIGRAPH : elle consiste a détecter automatiquement les informations peu intéressantes dans une image et à les supprimer en premier lors d'un redimensionnement.

> Le but de ce devoir est d'implanter cette méthode de redimensionnement d'image appelée *Seam Carving*. Vous trouverez une démonstration du résultat en vidéo sur internet à l'adresse suivante[1] :
> `http://www.youtube.com/watch?v=6NcIJXTlugc`

1 Gestion des images

Il paraît assez évident que vous aurez besoin de savoir manipuler des images. Vous devrez donc être capable de :

- lire et écrire des fichiers d'images ;
- afficher des images via le module `Graphics` d'OCaml ;
- définir votre représentation interne sur laquelle vous travaillerez, et qui sera l'intermédiaire entre les deux représentations précédentes.

Traditionnellement, une image est représentée par une matrice de couleurs, ces couleurs étant définis par leurs niveaux de rouge, de vert et de bleu. Dans ce devoir, vous travaillerez tout d'abord en monochrome (niveaux de gris), et donc la représentation interne des images sera simple.

1.1 Le format `ppm`

Parmi les formats d'image, le `ppm`[2] est un des plus simples. La version 5, qui concerne les images en niveaux de gris (la version 6 gère la couleur), s'appelle également `pgm`. Il peut s'obtenir grâce à l'utilitaire `convert` de la suite ImageMagick :

```
% convert mon_image.jpg mon_image.pgm
```

Examinons ensemble le résultat avec `% emacs mon_image.pgm`. Avant la marée d'hexadécimal se trouvent 3 lignes importantes :

```
P5
160 221
255
```

`P5` indique que c'est du `ppm` version 5, 160 est la largeur de l'image et 221 sa hauteur (en pixels). 255, c'est le nombre maximum utilisé comme valeur pour un pixel : tous sont compris entre 0 (noir) et 255 (blanc). On va donc plutôt regarder le fichier en mode hexadécimal ; pour cela appuyez sur 'echap' puis 'x', puis tapez « hexl-mode ». Ça devrait ressembler à :

```
00000000: 5035 0a31 3630 2032  3231 0a32 3535 0ac1  P5.160 221.255..
00000010: c1c1 c0c0 c0cb cbcb  c7c7 c7c7 c7c7 9b9b  ................
```

[1] Vous pourrez trouver d'autres informations sur internet ainsi que des implantations existantes. Ne cherchez pas à les comprendre (perte de temps) ou à les copier (perte de points).

[2] `http://netpbm.sourceforge.net/doc/ppm.html`

Au milieu vous voyez une ligne de 16 octets ; à droite la représentation ASCII (si c'est un caractère représentable). Notez que `P5`, 160, 221 et 255 sont écrits avec des vrais caractères (ex : '1', '6' et '0'). Ensuite, après 255, un retour à la ligne (code `0a`), puis est codée la matrice de l'image : chaque octet code la valeur d'un pixel ; les 160 premiers codent la première ligne de l'image, les 160 suivants la deuxième, et ainsi de suite pour les 221 lignes. Ce fichier doit donc au total contenir $160 \times 221 = 35360$ octets, plus l'entête de 15 octets (dans notre cas, ça peut changer selon la taille de l'image bien sûr). On peut vérifier avec `wc` :

```
% wc -c mon_image.pgm
35375 mon_image.pgm
```

Voilà le minimum de ce que vous devez savoir sur le format `ppm`. Vous avez à présent ce qu'il faut pour vous faire un module d'entrée/sortie minimal de `ppm` en OCaml. Pour aller plus loin allez lire le lien donné, et faites vous-même vos propres recherches.

1.2 Le module `Graphics`

Le manuel d'OCaml[3] et la page consacrée à ce module[4] vous seront utile, vous pourrez vous en inspirez pour votre représentation interne des images.

Il n'est pas obligatoire d'être capable d'afficher des images ; vous pouvez vous contenter d'un programme qui travaille uniquement en ligne de commande. Cependant, pour des facilités de débogage ou simplement parce que c'est rigolo de pouvoir voir ce qui se passe en temps réel sur l'image, il est conseillé de faire un module d'affichage.

2 Fonction d'énergie

Il est possible, en utilisant différents filtres d'images, d'obtenir des informations sur l'intérêt d'une partie d'image. Traditionnelement, on utilise le gradient comme moyen d'évaluation pour des questions de simplicité et donc de rapidité. Ainsi dans une photo, une partie avec un gradient élevé (ie une grande disparité de couleur, de luminosité, etc.) sera considérée comme contenant plus d'informations. Par exemple, le ciel bleu d'une photo aura un faible gradient, alors qu'un visage aura au contraire un gradient élevé. C'est ce gradient que l'on choisira pour la *fonction d'énergie*.

Dans cette section, on s'attachera uniquement au calcul de la fonction énergie ; la section suivante décidera des pixels à enlever (pour réduire une image) en fonction du gradient calculé ici, l'idée étant que les pixels avec une faible énergie seront les premiers à être effacés.

Une étape nécessaire d'un programme de seam carving est donc la création d'une fonction d'énergie calculant le gradient de l'intensité de l'image. En noir et blanc, l'intensité et le niveau de gris d'un pixel sont représentés de la même manière.

En mathématiques, le gradient d'une fonction à n variables est le champ vectoriel dont les composantes sont les dérivées partielles (selon chacune des variables). Dans le cas présent, nous sommes uniquement intéressés par la valeur de ce champ en chaque point. Si cette valeur est élevée, le point correspondant présente de fortes perturbations, c'est donc probablement un contour et donc un élément distinctif dans notre image.

[3] `http://caml.inria.fr/pub/docs/manual-ocaml/manual039.html`
[4] `http://caml.inria.fr/pub/docs/manual-ocaml/libref/Graphics.html#6_Images`

Votre programme devra utiliser une des fonctions d'énergie présentées ci-après. Il n'est pas nécessaire de les essayer toutes, mais prenez soin à ce qu'il soit facile d'utiliser une fonction d'énergie différente de la votre. Il est donc conseillé d'abstraire votre méthode.

2.1 Cas discret

Nous ne possédons pas ici de fonction continue et dérivable mais d'une image et donc seulement de données discrètes. Les dérivées sont donc des approximations en chaque point. Supposons donnée une image dont on a calculé l'intensité en chaque point dans la matrice I. Les valeurs du gradient de I sont calculées dans la matrice G ainsi :

- $dv_{i,j} = I(i, j-1) - I(i, j+1)$ et $dh_{i,j} = I(i-1, j) - I(i+1, j)$ (les dérivées verticales et horizontales au point (i, j)).
- alors $G(i, j) = \sqrt{dv_{i,j}^2 + dh_{i,j}^2}$

Prenez garde aux cas limites (bords de la matrice).

2.2 L'algorithme de Sobel

Pour être plus précis que dans la partie précédente, il est possible de calculer la dérivée discrète avec les 8 cases adjacentes. Les voisins diagonaux doivent être pondérés pour qu'ils aient moins influence. Ce genre d'opération, qui sert à beaucoup de filtre d'image peut s'effectuer grâce à une *matrice de convolution*.

2.3 Le détecteur de Harris

Il existe beaucoup de méthodes pour détecter les contours ou les coins d'une image dont celle d'Harris. Vous pouvez lire l'article de Wikipedia[5] pour plus d'informations.

3 Découper l'image

Pour réduire la taille d'une image, nous allons procéder incrémentalement en la diminuant d'un pixel (verticalement ou horizontalement) puis recommencer jusqu'au résultat voulu.

Dans la suite, on considérera un redimensionnement *horizontal*, on cherchera donc à retirer un pixel par ligne de l'image plusieurs fois jusqu'à la taille souhaitée.

Une première solution consiste à retirer le pixel de plus basse énergie de chaque ligne. Expliquez pourquoi cette solution est mauvaise, soit par des arguments convaincants, soit par l'expérience.

Une meilleure solution consiste à retirer la colonne d'énergie minimale, l'énergie d'une colonne étant la somme des énergies de ses pixels. Cette solution à l'avantage d'être efficace si plusieurs redimensionnements sont effectués séquentiellement. En effet, si l'on retire une colonne à une image, la matrice d'énergie ne sera modifiée que sur les deux colonnes adjacentes et il est donc très facile de trouver la nouvelle colonne de plus basse énergie.

Entre la première méthode, trop flexible, et la deuxième, un peu rigide, il existe une solution intermédiaire : trouver un chemin connexe d'énergie minimale qui traverse l'image de haut en bas. C'est cette méthode qu'on vous demande d'implanter.

[5] `http://en.wikipedia.org/wiki/Corner_detection`

Remarquez que le chemin de plus basse énergie reliant le haut de l'image au pixel (x, y) passe nécessairement par un des trois pixels du dessus. Vous pourrez donc utiliser la programmation dynamique pour trouver le chemin d'énergie minimale.

> Là encore, prévoyez d'avoir un code assez adaptable pour implanter facilement un autre algorithme de recherche d'une zone d'énergie minimum.

4 Une tonne d'améliorations à faire...

Optimisation : il est inutile de recalculer toute la matrice d'énergie après avoir enlevé un chemin de l'image, cherchez donc différentes façons d'accélérer les calculs. De même pour la recherche du chemin de plus basse énergie.

Couleur : cherchez maintenant comment gérer les formats `ppm` version 6 (couleur RGB). Pour l'énergie d'une image en couleur, vous pouvez prendre la luminosité des points, ou mieux si vous avez...

Interface graphique : utilisez les évènements souris et clavier et la sortie graphique d'Ocaml pour créer un petit logiciel de redimensionnement interactif.

Plus de *real-time* : votre programme est probablement trop lent pour que l'interface graphique soit vraiment interactive. Faites un pré-calcul de l'ordre dans lequel les pixels sont enlevés : annotez par '1' les pixels du premier chemin, par '2' ceux du deuxième, etc. Pour enlever le n-ième chemin, il suffit alors simplement de retrouver les pixels annotés par 'n'.

Protection et Suppression : la fonction d'énergie n'est pas idéale, la technique automatique de détection des contours marche généralement bien, mais on peut vouloir supprimer en priorité telle partie d'une image ou, au contraire, en garder une autre. Imaginez un système permettant à un utilisateur de spécifier les zones intéressantes ou non, à l'aide d'une image filtre ou de la souris.

Agrandissement : pour ne pas déformer les parties intéressantes, l'idée est de dupliquer le chemin d'énergie minimale, mais cette technique présente un problème majeur. Adaptez cette technique pour contourner cela.

5 Le code et le rapport

Votre code devra être abondamment commenté, indenté de façon intelligente et lisible (noms de fonctions, de variables, etc.).

La taille du rapport est libre, mais tentez d'être clair, précis et concis. Vous devez justifier vos choix (programmation, algorithmique, découpage du programme) et expliquez vos méthodes sans recopier du code. *Résumez* dedans ce que vous avez fait, ce que vous n'avez pas fait, ce que vous vouliez faire et comment, etc.

N'oubliez pas d'expliquer comment utiliser votre programme (ligne de commande, appel de fonction, options, ...).

Enfin, n'hésitez pas à nous envoyer un e-mail ou à venir nous voir pour nous poser des questions si vous butez sur un problème. C'est à ça que servent les e-mails que l'on met en entête de chaque TP :-).