TP Caml 5: Compression de niveaux de gris

lionel.rieg@ens-lyon.fr

9 décembre 2008 & 6 janvier 2009

1 Description théorique

1.1 Problème

On se donne une image en M niveaux de gris de $h \times l$ pixels représentée par une matrice d'entiers de taille $h \times l$ donnant le niveau de gris de chaque pixel. Notre but va être de réduire le nombre de niveaux de gris utilisés à k. Mais on veut le faire intelligemment de façon que la perte de qualité soit la plus faible possible. On va donc choisir soigneusement quels niveaux de gris garder.

Si on note $a_1, \ldots, a_k$ la palette des niveaux de gris conservés, on va remplacer le niveau de gris $v(p)$ de chaque pixel p par le a_i le plus proche, *i.e.* le niveau de gris a_i qui minimise l'erreur $|a_i - v(p)|$. On note $\varepsilon_{a_1,\ldots,a_k}(p)$ cette erreur d'approximation au pixel p.

Afin de limiter la perte en qualité, on va s'efforcer de trouver les k niveaux de gris qui minimisent $\varepsilon^t(a_1, \ldots, a_k)$ l'erreur totale sur l'image, définie comme

$$\varepsilon^t(a_1, \ldots, a_k) = \sum_{p \text{ pixel}} \varepsilon_{a_1,\ldots,a_k}(p)$$

1.2 La solution par programmation dynamique

Au vu de la définition de l'erreur totale, il est évident que seule la fréquence d'apparition de chaque niveau de gris est importante. On va donc utiliser la fréquence de chaque niveau de gris dans l'image (en fait, ce ne seront pas les fréquences mais plutôt les occurrences pour ne travailler qu'en entiers) plutôt que l'image elle-même. Ce sera facile à calculer : un simple parcours de l'image.

Pour résoudre ce problème, on va utiliser la programmation dynamique. Afin de réduire la dimension du problème (et par conséquent l'espace de solution à parcourir par programmation dynamique), on va supposer dans un premier temps que le dernier niveau de gris M de l'image originale est utilisé dans la palette à conserver. La formule de récurrence que l'on va utiliser donne donc l'erreur totale d'une répartition de k niveaux de gris dans une palette qui s'étale de 0 (noir) à $n < M$ (blanc) lorsque le dernier niveau de gris choisi est le dernier disponible, *i.e.* lorsque $a_k = n$. On ignore les pixels de niveaux de gris supérieurs à n (*i.e.* les nuances claires).

► **Question 1** Soit a_j et a_{j+1} deux niveaux de gris consécutifs dans la palette retenue. Calculer l'erreur $\delta(a_j, a_{j+1})$ commise en approximant par a_j ou a_{j+1} les pixels dont la couleur originale est comprise entre a_j et a_{j+1}.

► **Question 2** En notant $\text{Err}_k(n)$ l'erreur minimale de compression avec k niveaux de gris $a_1, \ldots, a_k$ compris entre 1 et n (avec $a_k = n$) sur les pixels de couleurs originales comprises également entre 1 et n, déterminer la relation de récurrence qui lie $\text{Err}_k(n)$ et les $\text{Err}_{k-1}(j)$. Quels sont les cas d'initialisation ?

2 Implémentation

► **Question 3** Écrire une fonction

```
interval_cost : int vect → int → int → int
```

qui calcule l'erreur engendrée par deux niveaux de gris consécutifs de la palette retenue, selon la formule calculée à la question 1. Ainsi, interval_cost freq a_i a_j calculera l'erreur commise en approchant par a_i ou a_j les pixels de couleurs comprises entre a_i et a_j sur une image dont le tableau de fréquence est donné par freq.

► **Question 4** Écrire l'algorithme de calcul de l'erreur totale minimale lorsque le dernier niveau de gris de la palette retenue est le dernier niveau de gris original (*i.e.* $a_k = n$) en utilisant la relation de récurrence de la question 2. Il sera de type

```
optimal_loss : int vect → int → int vect vect
```

► **Question 5** On veut obtenir une palette qui réalise le minimum de perte pour faire la compression. Modifier donc l'algorithme précédent pour garder trace de la case à la ligne précédente qui réalise l'optimum en la case courante. Son nouveau type sera

```
mem_optimal_loss : int vect → int → (int * int option) vect vect
```

► **Question 6** Terminer l'algorithme de calcul des niveaux de gris optimaux en omettant l'hypothèse selon laquelle le dernier niveau de gris conservé est le dernier disponible. Il renverra la liste des niveaux de gris d'une palette optimale (elle peut ne pas être unique).

3 Utilisation de l'algorithme

À présent que l'on possède l'algorithme de meilleur échantillonnage, on va s'en servir pour modifier de vraies images.

Pour commencer, il nous faut lire une image en niveaux de gris. Pour rester simple, on va utiliser le format PGM (Portable Grey Map), dont les spécifications (légèrement modifiées) sont :

1. sur la première ligne, le *nombre magique* « P2 » qui permet d'identifier le type du fichier (en plus de son extension)
2. sur la deuxième ligne et séparées par une espace [1], les largeur l et hauteur h de l'image
3. sur la troisième ligne, la valeur maximum M des niveaux de gris (0 = noir et M = blanc)
4. à partir de la quatrième ligne, h lignes de l nombres séparés par des espaces et compris entre 0 et M qui dénotent le niveau de gris de chaque pixel de l'image correspondant à la position dans cette matrice.

Pour manipuler les fichiers, on va se servir des types in_channel et out_channel de Caml qui autorisent respectivement à lire et écrire des fichiers mais pas les deux à la fois. Noter que lire ou écrire « consomme » le fichier, *i.e.* on ne peut relire ou récrire. Pour cela, on utilisera les fonctions suivantes de la bibliothèque standard :

- open_in : string →in_channel qui ouvre un fichier en lecture à partir de son nom
- input_line : in_channel →string qui permet de récupérer la ligne suivante d'un fichier ouvert en lecture
- close_in: in_channel →unit qui permet de fermer un fichier ouvert en lecture à la fin de son utilisation
- open_out : string →out_channel, output_string : out_channel →string →unit et close_out : out_channel →unit qui sont les pendants pour l'écriture

► **Question 7** Écrire une fonction

```
find_word : string → int → int * int
```

qui renvoie la position de début et la longueur du premier mot dans la chaîne de caractère donnée en premier argument à partir de la position donnée en second argument. Les espaces avant ce mot doivent être ignorées. Par exemple, si find_space s i vaut (k, l), cela signifie que le premier mot de s depuis le caractère s.[i] commence à l'indice k et comporte l caractères.

► **Question 8** Écrire une fonction get_header qui récupère l'en-tête d'une image PGM, *i.e.* qui prend en argument un fichier PGM ouvert à la seconde ligne et renvoie les dimensions de l'image et la valeur maximale des niveaux de gris (qui correspond au blanc).

► **Question 9** Écrire une fonction

```
frequency_array : in_channel → int → int → int → int vect
```

qui lit une image et génère le tableau de fréquence des niveaux de gris de l'image. Ses arguments sont :

- le fichier de l'image ouvert en lecture après l'en-tête (*i.e.* au début de la matrice)
- les dimensions de l'image (largeur puis hauteur)
- le niveau de gris M d'un pixel blanc

► **Question 10** Écrire une fonction qui, en appliquant l'algorithme défini à la partie précédente, transforme un tableau de fréquence en un tableau qui associe à chaque niveau de gris original celui de la palette qui le remplace.

► **Question 11** Écrire une fonction

```
compress : string → int → string → unit
```

qui compresse une image (à partir de son nom de fichier) en niveaux de gris en une image (en donnant le nom du fichier de sortie) avec moins de niveaux de gris. On n'oubliera pas de vérifier que le nombre magique est correct et d'écrire l'en-tête dans le fichier de sortie pour que le fichier compressé soit lisible.

1. En typographie, *espace* est un mot féminin.