BUT-Info.3 2023-2024
Module "3 en 1"
Projet n°2

Format de compression d'image numérique

Compression avec pertes

Adaptation pour la compression avec pertes

Sur des images de type photographique, le traitement proposé ici produit en sortie des fichiers souvent plus volumineux que les originaux. C'est normal : de telles images ne contiennent généralement que très peu de blocs parfaitement uniformes. Pour améliorer les performances, il faut introduire des pertes dans le processus de compression.

Un processus de compression est dit *avec pertes*, ou *destructif* s'il ne permet pas de reconstituer les données à l'identique. L'intérêt de telles techniques est d'améliorer les taux de compression en acceptant de perdre une part de l'information originale jugée non indispensable.

Dans le cas présent, le processus de compression est basé sur la recherche de blocs *uniformes* : plus ils sont nombreux et de grande taille, plus le processus sera performant.

On peut alors s'intéresser aux blocs *presques uniformes* pour les rendre *parfaitement uniformes* (dégradation). Pour cela, lors de la construction du `QuadTree`, il faudra évaluer, en plus de la *moyenne* des quatre voisins, un paramètre de *variance* (noté ν - dispersion des valeurs d'un ensemble autour de leur valeur moyenne) :
☞ plus la variance est faible, plus les valeurs sont proches.

Mathématiquement, la variance d'une série statistique $(x_i)_{0\leq i<n}$ de moyenne m_x est donnée par :

$$\nu_x = \frac{1}{n}\sum_{i=0}^{i<n}(m_x - x_i)^2 \quad (\textit{"moyenne des carrés des écarts à la moyenne"})$$

Ici, pour un nœud interne (x), cela sera fait simplement en mesurant la valeur absolue de la différence entre la valeur du noeud m_x et la valeur de chacun de ses fils $(m_{s_i})_{0\leq i<4}$.

Afin de tenir compte de ce qu'il se passe "sous" le neud courant x, on incluera dans ce calcul les variances des fils $(\nu_{s_i})_{0\leq i<4}$. Ainsi, un noeud "riche" en information (variance élevée) élévera la variance de son noeud père, même si sa moyenne est strictement égale à celle de ses noeuds frères.

☞ $$\nu_x = \sum_{i=0}^{i<4}\left(\nu_{s_i} + (m_x - m_{s_i})^2\right)$$

Au final, un noeud interne (x) sera déclaré "presque-uniforme" si **tout** le sous-arbre qu'il recouvre peut-l'être aussi, c'est à dire si ses 4 noeuds fils ont également été déclarés uniformes[a] et si sa variance ν_x est inférieure à un seuil τ (paramètre du codeur).

Son champ `u` est mis à `1` et son champ ϵ est forçé à `0`) et tout le sous-arbre situé en dessous sera ignoré lors du codage.

[a] *parcours en profondeur suffixe*

Cette valeur de variance devient donc une donnée constitutive du `QuadTree` sous la forme d'un champ `var` (de type `double`) intégré à la struture de `noeud`. Elle est calculée à la volée, lors du parcours récursif *suffixe* de remplissage de l'arbre.

Le cas $(\tau = 0)$ correspond bien sûr au cas sans pertes mais si le seuil τ est trop important, cela se traduira par un effet de *pavage* dans l'image décodée.

Mise en oeuvre et dispositif de "simulation"

Concrètement, un bon dosage, pour une première version, consiste à définir une fonction de filtrage parcourant l'arbre en profondeur de la racine jusqu'aux feuilles, avec :

- en valeur initiale de seuil (racine) : $\tau_0 = \alpha * \nu_0$ où ($\alpha \geq 0.$) est un paramètre ajustable du codeur.
- en ajustant τ à chaque niveau `k>0` de l'arbre par $(\tau_k = \alpha * \tau_{k-1})$.

☞ α est **le** paramètre qui détermine le rapport coût/qualité

- le cas $\alpha = 0.$ correspond au cas *lossless*.
- pour des images photographiques, la segmentation par `QuadTree` commence à produire des effets notables à partir de $\alpha \geq 0.5$.
- à partir de $\alpha \geq 1.$ les dégradations deviennent vraiment fortes.

De son côté, le décompresseur n'a pas à savoir que le codage a introduit des pertes : il formera des blocs parfaitement uniformes.

Remarque : l'introduction d'un processus de *dégradation* dans le schéma de compression n'a **aucune** incidence sur le schéma d'encodage (les fonctions d'écriture/lecture restent les mêmes) et n'a **aucun** impact sur le décodeur.

☞ en revanche, c'est sur ce point précis (filtrage du `QuadTree`) que se joue la différence entre un *bon* encodeur et un moins bon : le *bon* encodeur réussira à maximiser les pertes tout en minimisant l'impact sur la qualité de l'image encodée.

Le processus de gestion des pertes proposé ci-dessus n'est clairement pas très bon. On peut, assez facilement, faire **beaucoup** mieux.

☞ ce sera donc un élément fondamental à travailler avec soin puisque c'est là que se fera la différence entre deux `CoDec`.

☞ rien n'est imposé sur ces aspects, tant que cela n'impacte ni la structure du format ni le décodeur.

Pour pouvoir tester et visualiser ces modes de dégradation, il sera nécessaire de disposer d'un environnement de "simulation", c'est-à-dire avoir la possibilité d'évaluer, graphiquement, l'impact d'un algorithme ou d'un réglage sur la qualité visuelle de l'image.

Certaines fonctionalités (filtrage/écriture) pourront ainsi être *émulées* en mode test pour évaluer les taux de compressions obtenus. Cela peut se faire sans modifier les structures et interfaces mais nécessite de recharger le `QuadTree` depuis l'image originale (ou disposer d'une "copie" du `QuadTree`).