Yannis ANDRE

Emeric BARBOT

Côme ALEXANDRE

Rapport pour le problème scientifique et informatique

Une image contenant texte

Description générée automatiquement

Voici la première image que l’utilisateur voit lorsqu’il exécute notre programme. Il lui est demandé quelle image il choisit pour tester les transformations que l’on peut faire.

Une image contenant texte

Description générée automatiquement

Ensuite, il lui est demandé quel traitement de l’image il veut effectuer, sachant que chaque traitement est possible pour chaque image, sauf le codage d’Huffman, qui n’est disponible que pour l’image de femme et le carré de pixels noirs et blancs, car elle prend un temps trop long pour les grandes images et les images colorées. Il n’est par ailleurs pas conseillé de décoder deux images cachées l’une dans l’autre si l’image ne contient pas deux images.

Si l’utilisateur choisit l’option 6, il est ensuite en face du menu suivant :

Une image contenant texte

Description générée automatiquement

Et s’il choisit l’option 8, il lui est demandé quelle image il veut cacher dans la première image qu’il a choisie. Enfin, il lui sera demandé de donner un nom de fichier à la nouvelle image qui sera créée, qui ne pourra pas être un nom utilisé par les images de base (celles du premier menu).

Nous allons maintenant vous décrire les méthodes permettant d’effectuer les traitements demandés par l’utilisateur.

Passage en nuances de gris : Pour chaque pixel, on calcule la moyenne de la somme des valeurs du pixel en rouge, vert et bleu, ce qui va correspondre à une nuance de gris si on l’attribue à chaque couleur.

Image en noir et blanc : Pour chaque pixel, on calcule aussi la moyenne de la somme des trois couleurs du pixel. Ensuite, si cette moyenne est inférieure à 128, on colorie le pixel en noir, et sinon on le colorie en blanc. On fait cela car on arrondie la valeur de la moyenne au plus proche de 0 ou de 255 pour chaque couleur, 0 représentant noir et 255 représentant blanc.

Rotation de l’image : L’utilisateur choisit un angle. On commence par calculer le cosinus et le sinus, en soustrayant 90 à l’angle afin d’effectuer la rotation dans le bon sens. Puis on cherche autour de quel point la rotation va s’effectuer. Puis on multiplie les coordonnées par la matrice de rotation, et pour les pixels en dehors de l’image, on les remplit par un pixel noir.

Agrandissement de l’image : On demande le coefficient d’agrandissement à l’utilisateur. On multiplie les dimensions par le coefficient d’agrandissement, et on calcule le nombre d’octets nécessaires pour la création de la nouvelle image. Ensuite, les nouveaux pixels dus à l’agrandissement vont prendre les valeurs de leur plus proche voisin, calculé en divisant l’indice des lignes et colonnes par le coefficient d’agrandissement.

Filtres de convolution : L’utilisateur choisit quel filtre il veut. Pour pouvoir appliquer les matrices de convolution, nous avons eu quelques problèmes, car nous ne réussissions pas à faire la multiplication, que nous avons réglés en ajoutant un carré de pixels noirs autour de l’image afin de pouvoir appliquer la matrice de convolution. On parcourt les 8 sommets adjacents d’un pixel, puis on additionne les valeurs des couleurs multipliées par le coefficient correspondant au motif, et on insère le nouveau pixel obtenu dans notre matrice résultat.

Fractale de Mandelbrot : On choisir qu’à partir de 100 itérations, le point fait partie de l’ensemble de Mandelbrot. Puis, pour chaque pixel on définit la partie réelle et imaginaire de z0 ou c notre nombre complexe d’origine, puis, on fait une itération ou on remplace ce nombre par Zn^2 +z0, tant que son module est inférieur à 2 ou que le nombre d’itérations est inférieur à 100. Les pixels dont le nombre d’itérations sont proches de 100 seront coloriés en blanc, et les autres en noirs.

Cacher une image dans une autre : Le principe est de ne conserver que les bits à gauche des composantes rgb d’une image, et d’y ajouter les bits à gauche de l’image à cacher, car ils correspondent aux bits aux poids les plus forts, et qui donc contiennent l’essentiel de l’information. En pratique, pour chaque pixel on prend les 4 premiers bits de l’écriture binaire de la composante rgb du pixel de la première image, et on y ajoute les 4 premiers bits de l’écriture binaire de la composante rgb de l’image à cacher. Puis on convertit le nombre ainsi obtenu en décimal pour chaque composant rgb.

Décoder deux images cachées l’une dans l’autre : Même protocole mais à l’envers, on décompose l’octet de chaque composante rgb pour chaque pixel, et la première moitié sera pour la première image, et la seconde pour l’image supposément cachées.

Huffman : Pour modéliser le codage d’Huffman, on crée une classe arbre et une classe nœud. Un nœud est composé des informations suivantes : son parent (s’il en a un), ses deux fils (s’il en a, a quel pixel il correspond et sa fréquence). L’arbre est composé des informations suivantes : un nœud dont sa fréquence correspond au nombre de pixels total de l’image, à dictionnaire qui pour chaque pixel associe une fréquence, et une liste de nœuds correspondant aux feuilles de l’arbre. Tout d’abord, pour chaque pixel différent on associe une fréquence. Puis on associe un nœud à chaque pixel différent, et on récupère sa fréquence. Ensuite, tant qu’il y a plus d’un nœud, on additionne les nœuds avec les plus faibles fréquences entre eux, et on additionne les fréquences des nœuds qui le composent. A la fin, nous n’avons plus qu’un nœud avec une fréquence égale au nombre de pixels de l’image. Pour compresser, on redéroule l’arbre jusqu’à avoir juste les premiers nœuds correspondant aux pixels différents, et le code pour chaque pixel correspondra au chemin qu’on fait jusqu’à l’atteindre, en partant des parents, et en codant 0 si le fils est à gauche, et 1, si le fils est à droite. Pour décompresser, on prend le chemin donné par la compression, et on descend l’arbre jusqu’à trouver le pixel qui correspond à la fréquence. Ensuite, on donne ses valeurs.

Innovations :

Rétrécissement : fonctionne avec la fonction pour l’agrandissement.

Floutage par pixellisation : On prend un nombre de pixels correspondant au taux de pixellisation, et on crée un bloc, tel que tous ces pixels aient la même couleur. Pour déterminer quelle couleur, on fait la moyenne des couleurs rgb de tous ces pixels pour obtenir un pixel, dont tous les pixels du bloc prendront les valeurs rgb.