Projet Matlab : un logiciel de cryptage

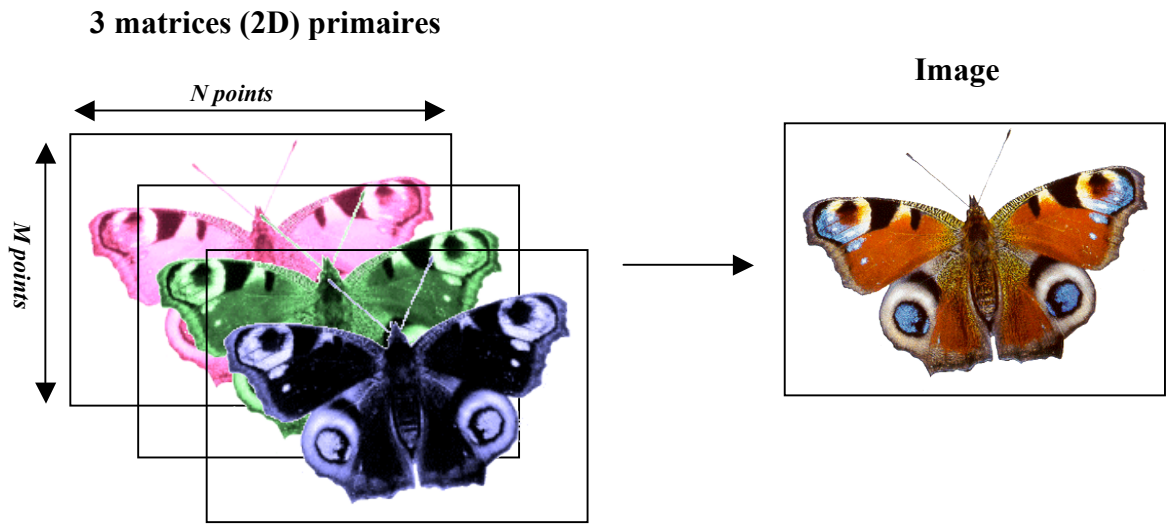
La stéganographie (du grec *steganos :* couvert et *graphein :* écriture) consiste à dissimuler une information au sein d'une autre à caractère anodin, de sorte que l'existence même du message caché passe inaperçue.
Alors qu'avec la cryptographie, la sécurité repose sur le fait que le message ne sera sans doute pas compris, la stéganographie, s'appuie sur le fait que le message ne sera pas détecté.

Ce projet vous propose de mettre en application l'une des méthodes les plus basiques, en termes de stéganographie. L'idée est de prendre une image et de la modifier de manière imperceptible afin d'y dissimuler l'information à transmettre, en l'occurrence une autre image binaire (composée de deux niveaux : 0 et 1) contenant du texte.

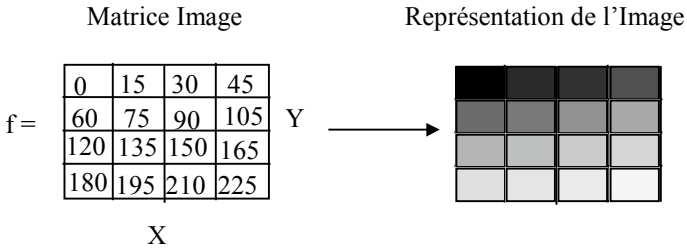
I. Quelques notions essentielles

1- Codage des images sous matlab

Une image n'est autre que l'extension d'un signal à 2 dimensions. Une image couleur est la superposition de 3 composantes de base (Rouge Vert et Bleu). Sous Matlab une telle image peut être codée par une matrice tridimensionnelle. Elle correspond à la mise en cascade des 3 matrices (2D : N*M) correspondant aux 3 composantes primaires (Rouge Vert et Bleu). Chacune de ces 3 matrices primaires (aussi appelées « plans couleur ») contient le niveau de couleur pour chaque point de l'image considérée. En général chaque niveau de couleur est codé entre 0 (canal colorimétrique éteint) et 255 (canal colorimétrique maximum) ce qui correspond à 255^3 = 16 581 375 combinaisons de couleurs possibles.



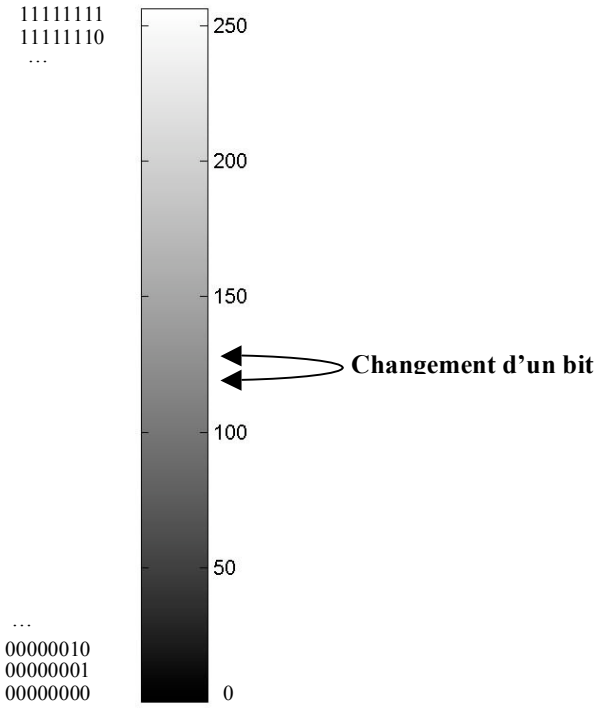
Dans le cas particulier des images en noir et blanc, les 3 composantes de chaque point de l'image considérée (pixel) sont égales. Nous avons donc 256 niveaux de gris (de 0 pour le noir à 255 pour le blanc, en passant par un gris moyen pour 128).



<u>Rq :</u> Lorsque l'on charge une image sous matlab, la matrice correspondante est de type uint8 (codage sur 8 bits non signés, autrement dit : 0-255). Pour simplifier les calculs et pouvoir utiliser toutes les fonctions matlab sur ces données, il est préférable de les convertir en réels (commande « double »), puis après traitement les reconvertir en entier codé sur 8 bits non-signé avant de sauvegarder en format bmp (commande «uint8 »).

2- Une technique de stéganographie

La technique de base, dite ***LSB*** (pour ***L****east* ***S****ignificant* ***B****it* ou *bit de poids faible*, en français) consiste à modifier le(s) bit(s) de poids faible des pixels codant l'image. En effet, supposons que le niveau des couleurs soit codé sur 8 bits (256 niveaux, comme nous l'avons vu précédemment), le fait de modifier le ou les bits de poids faible entraînera un changement de couleur presque imperceptible pour le pixel considéré.



Cette technique de stéganographie très basique s'applique tout particulièrement au format d'image BMP, format sans compression destructive, avec codage des pixels sur 3 octets (3*8 bits) comme énoncé ci-dessus.

II. Travail à réaliser

Le logiciel que vous allez constituer, va donc permettre, à l'aide de la méthode de sténographie présentée ci dessus, de crypter une information cachée dans une image. Cette information sera sous la forme d'une image binaire (en noir et blanc) de texte, comme dans l'exemple ci-dessous :

L'orang-outan appartient à la famille des pongidés qui comporte deux sous-espèces : pongo pygmaeus pygmaeus qui vit à Bornéo, et pongo pygmaeus abelii que l'on trouve à Sumatra. Sa taille varie entre 110 cm pour la femelle et 140 cm pour le mâle, tandis que son poids oscille entre 30 et 90 kg selon le sexe. La gestation dure 245 jours, et il n'y a qu'une seule naissance tous les 4 ou 5 ans. Son espérance de vie atteint une quarantaine d'années. Peu fourni sauf sur les épaules où il peut atteindre 50 cm de long, son pelage est d'un brun plus ou moins roux. Il occupe exclusivement les forêts ombrophiles de Malaisie (provinces de Sabah et Sarawak), du Sultanat de Brunei Darrussalam, et d'Indonésie (Kalimantan et Sumatra). En malaisien, son nom signifie « homme de la forêt »

A droite, l'image dans laquelle est cryptée le texte (mis sous la forme d'une image) de gauche

Dans cet exemple (dont nous vous fournirons l'image), le texte est codé dans le dernier bit du plan « rouge » de l'image de droite ; un bit à 1 correspondra à un pixel blanc et un bit à 0 un pixel noir de l'image de texte. Il suffit donc d'extraire la matrice (composée de 0 et 1) de l'image de ouran-outan pour retrouver le texte (sous la forme d'une image noir et blanc).

Afin de constituer ce logiciel, vous allez **procéder par étapes** :

- la première constituera en le codage de méthodes simplifiées de cryptage et décryptage, où l'information cachée ne sera cryptée que dans un seul plan (à la manière de l'exemple de l'oran-outan) et où les images cachées seront crées à l'aide d'un logiciel externe à Matlab (MS Paint).

- la seconde introduira la notion d'une clef de cryptage (relativement simple) qui sera indispensable pour décrypter correctement l'image. Seules les personnes connaissant cette clef pourront décrypter simplement l'image.

- La troisième étape consistera à ajouter la possibilité de créer une image de texte à partir de Matlab (sans passer par un logiciel externe).

Il sera important de créer une **interface utilisateur transparente** (c'est-à-dire que des personnes ne connaissant quasiment rien en Matlab puissent utiliser), en utilisant notamment les fonctions d'entrées-sorties (« input », « disp »,…) ainsi que des menus à boutons.

1. Scripts simplifiés de cryptage et décryptage

Lors de cette étape, il ne s'agit pas encore de créer le logiciel final, mais simplement pour vous de mettre en pratique, et ainsi de maîtriser, les méthodes de cryptage et décryptage.

Tout d'abord, vous devez prendre une image quelconque (cherchez la sur internet par exemple) que vous devez convertir en format bmp (par exemple, à l'aide de MS Paint ou même de Matlab – avec « imread » et « imwrite »). Nommons-la « image.bmp » pour faciliter la suite des explications. Ensuite à l'aide de MS Paint, créez une image de même taille (cf. l'option « attributs » de MS Paint) que votre image et toute blanche. Dans cette seconde image, à l'aide de MS Paint, écrivez quelque chose en noir. Sauvegardez en format bmp (monochrome) sous un autre nom (par exemple « imagetexte.bmp »).

Ensuite, créez deux scripts « cryptage.m » et « decryptage.m » :

- Le premier doit charger les deux images, puis remplacer le bit de poids faible du plan « rouge » de l'image « image.bmp » avec l'information contenue dans l'image « imagedetexte.bmp ». Puis sauvegarder l'image obtenue sous format « bmp » (sous le nom « imagecrypte.bmp »).
- Le second doit charger l'image « imagecryptee.bmp » pour en extraire du plan « rouge » l'information cryptée.

Nous vous rappelons que la valeur d'un pixel est codée sur 8 bits, selon la relation suivante : valeur = bit0 + bit1*2^1 + bit2*2^2 + bit3*2^3 + … + bit7*2^7.
Par exemple : 173 = 1+0*2+1*4+1*8+0*16+1*2^5+0*2^6+1*2^7 -> 10101101

Vous pouvez utiliser la fonction **imagesc** de Matlab pour visualiser les différentes images.

2. Clef de cryptage

Précédemment, vous avez codé l'information cachée dans un seul plan (le « rouge ») de l'image couleur. Cependant, toute personne au fait de la technique serait en mesure de décrypter votre image (et donc de lire l'information cachée). A présent, vous allez introduire une clef de cryptage : seule la personne qui connaît aussi la clef sera en mesure de décrypter votre image.

Dans cette partie, la clef consistera en une suite de 1, 2 et 3, correspondant aux plans couleurs où sera codée l'information : ainsi si la clef est 231321, le premier pixel de l'image « imagedetexte.bmp » sera codée dans le plan 2 (i.e. « vert »), le second pixel sera codé dans le plan 3 (i.e. « bleu »), le troisième dans le plan 1, le quatrième dans le plan 3, le cinquième dans le plan 2, le sixième dans le plan 1. Les pixels suivants seront codés en répétant la clef ; le septième sera codé dans le plan 2, le huitième dans le plan 3, etc.

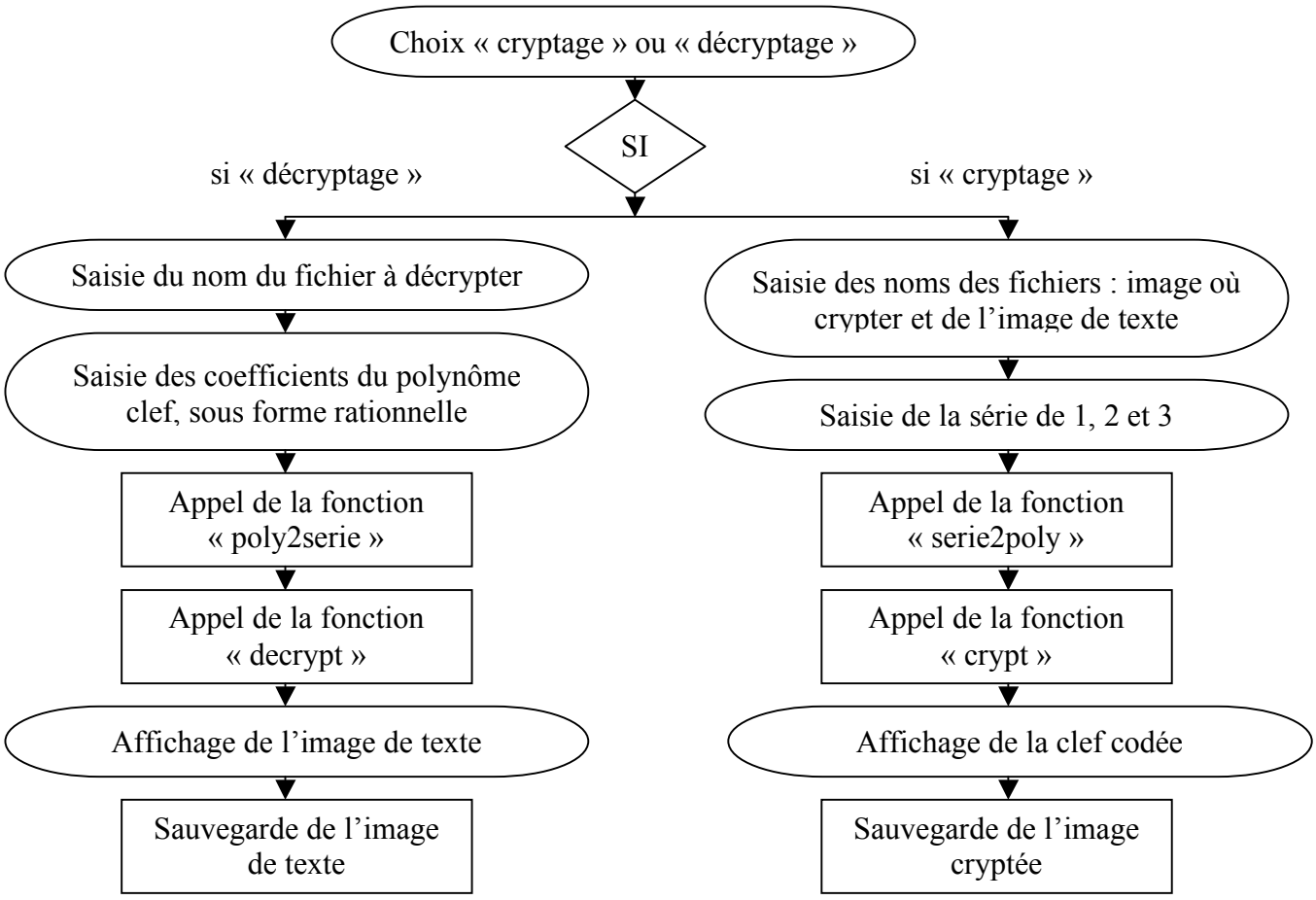
Pour compliquer encore un peu, nous allons coder cette clef en la traduisant sous forme des coefficients (sous forme rationnelle) du polynôme passant exactement par les points correspondants. La clef 231312 sera ainsi traduite par les coefficients du polynôme passant par les points de coordonnées (1,2), (2,3), (3,1), (4,3), (5,1) et (6,2). Il s'agit donc d'un polynôme d'ordre 5 : 1/4*x5-35/8*x4+86/3*x3-693/8*x2+1417/12*x-54=0. La clef codée finale sera la concaténation des numérateurs et des dénominateurs :

1 -35 86 -693 1417 -54 4 8 3 8 12 1

Au niveau de la forme, le programme devra incorporer une interface utilisateur transparente et se décomposer en plusieurs scripts et fonctions. Un **script principal** devra demander à l'utilisateur les informations nécessaires au fonctionnement et lancer les différentes fonctions nécessaires :

- en premier lieu, il demandera si l'utilisateur veut crypter ou décrypter.
- en fonction de la réponse de l'utilisateur, il demandera les noms des fichiers images à crypter ou à décrypter.
- puis il demandera la clef de cryptage soit sous la forme d'une série (dans le cas du cryptage) soit sous la forme des coefficients du polynôme (dans le cas du décryptage).
- Il lancera alors une fonction de traduction de la clef (série->polynôme, « serie2poly.m, ou polynôme-> série, « poly2serie.m »,)
- Puis il lancera une fonction de cryptage (« crypt.m ») ou de décryptage (« decrypt.m »).
- Enfin, il en affichera le résultat (l'image résultante, ou de la clef codée) et le sauvegardera dans un fichier bmp.

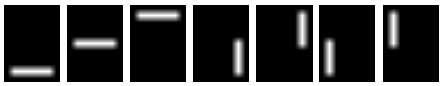
Ceci peut être mis sous la forme de l'organigramme suivant :



3. Création sous Matlab d'une image de texte

Un des points faibles du programme développé est la nécessité de faire appel à un programme externe (MS Paint) pour créer l'image de texte. Dans cette dernière étape, vous allez coder une fonction pour créer l'image de texte selon un texte saisi par l'utilisateur.

Afin de réaliser cela, chaque caractère du texte saisi donnera lieu à une petite matrice (de taille 11x8) de 0 et 1 dessinant le caractère. En fait, chaque matrice sera issue de l'assemblage de plusieurs éléments de base (composés de barres verticales et horizontales), à la manière d'un affichage digitale d'une montre. Nous limiterons les caractères aux seuls caractères numériques ('0','1','2','3','4','5','6','7','8' et '9'). Tous autres caractères donnera lieu simplement à une matrice nulle (toujours de taille 11x8), correspondant à un « espace » dans le texte. Ceci nous permet de nous limiter dans le nombre d'éléments de base. Ceci sont au nombre de 7 :



Par exemple, pour obtenir le caractère '9', nous combinons :

Il suffit donc de traduire tous les caractères numériques saisis par l'utilisateur sous cette forme matricielle et de les insérer (chaque caractère à la suite de l'autre) dans une matrice de la même taille que l'image où crypter.

Attention, chaque caractère, traduit sous la forme de matrice, aura une taille de 11x8 pixels. Dans une image de taille 480x640 pixels, le maximum de caractères possibles est donc de 3440 (80 caractères par ligne et 43 caractères par colonne). Vous devrez tenir compte de cette limitation. De même, les dimensions de l'image ne sont pas forcément des multiples de 11 ou de 8.

III. <u>Améliorations facultatives</u>

Bien que cela ne soit pas requis, vous pouvez, si vous disposez encore de temps, améliorer encore votre programme :

- en incluant d'autres caractères (les caractères alphabétiques, les ponctuations,…). Cela nécessitera de définir d'autres éléments de base (des barres diagonales…).
- en incluant votre programme dans une boucle infinie qui ne s'arrêtera que si vous faîtes le choix de quitter le programme (option à ajouter dans le choix). Ceci permettra de répéter plusieurs fois les opérations de cryptage ou de décryptage sans relancer à chaque fois le programme.
- en compliquant le cryptage ; actuellement, seul le bit de poids le plus faible (sur un des 3 plans couleurs) sert au cryptage. Il est possible de répartir aussi sur d'autres bit de poids faibles (un parmi les 4 bits les plus faibles) selon une seconde clef de cryptage.

Vous pouvez aussi proposer vos propres améliorations. Toutes ces améliorations sont facultatives.

IV. Annexe : fonctions Matlab utiles

Dans cette annexe, nous présentons une liste de fonctions Matlab qui pourraient vous être éventuellement utiles (en plus, de celles que vous connaissez déjà) dans ce projet. N'hésitez pas à lire l'aide Matlab (à l'aide de la commande « help ») pour comprendre comment elles fonctionnent et les utiliser.

floor : renvoie la partie entière d'un nombre

imagesc ou **image** : affiche une image.

imread : permet de charger une image

imwrite : permet de sauvegarder une image

colormap : attribut aux différents pixels de l'image la couleur correspondante.

rem : renvoie le reste de la division entière

reshape : permet de réorganiser les lignes et colonnes d'une matrice (changement de taille)

rat : renvoie la forme rationnelle d'un nombre (numérateur et dénominateur)