TÉLUQ

TRAVAIL PRATIQUE 1

Compression d'images fixes

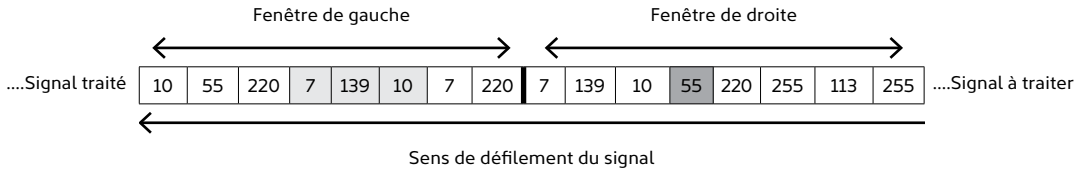
Présentation

L'objectif de ce travail pratique est de **mettre en application l'algorithme de compression sans perte LZ77**. Notons qu'il existe de multiples variantes de cet algorithme. Toutes les versions reposent sur la même procédure de codage, mais se différencient par le format de représentation du code. Dans le cadre de ce TP, on s'intéressera à implémenter la version originale de l'algorithme décrite ci-dessous, car elle est plus adaptée pour un codage en MATLAB. Cette procédure de codage sera appliquée pour compresser une image à niveaux de gris.

L'algorithme LZ77 (Lempel et Ziv, 1977)

Tous les algorithmes Lempel-Ziv se basent sur l'étude de répétition des symboles. La version originale de LZ77 proposée par Lempel et Ziv en 1977 procède de la manière suivante :

On fait passer le signal à coder dans une fenêtre de taille N divisée en deux parties de tailles n1 et n2. Les symboles sont ensuite défilés de droite à gauche. Si le début de la partie droite se trouve dans la partie gauche, on code la position de la sous-chaîne et sa longueur. Voici une illustration :



Le symbole courant est à droite du trait gras et la section de gauche a déjà été codée. On compare le début de la partie droite à la partie gauche en cherchant des séquences identiques. À l'issue de la recherche de correspondance, on récupère un couple (position, longueur), qui pourrait être par exemple (0, 0). Dans l'exemple précédent, on obtient (4, 3) : trois symboles à partir du numéro 4 (le '10' de gauche dans la fenêtre de gauche est numéroté 1). Cela signifie que le symbole à la position n1+longueur+1 (le '55') n'est pas inclus dans la chaîne de correspondance trouvée. On l'ajoute au codage pour obtenir le code (4,3, '55'). La sortie de l'algorithme LZ77 est ainsi un triplet (position, longueur, suivant) pour chaque symbole traité, où :

position : indice de 1 à n1 de la partie de droite trouvée à gauche

longueur : la longueur de la série de symboles trouvée à gauche

suivant : le premier symbole de droite qui n'a pas été trouvé à gauche

Un autre exemple

Soit le signal d'entrée : 00101021021021240101021 00...

n1=9 (taille de la fenêtre de gauche).

N=18 (taille de toute la fenêtre glissante).

On initialise l'algorithme en remplissant la fenêtre de gauche (le dictionnaire) par des zéros et on charge dans la fenêtre de droite (fenêtre de lecture) les 9 premiers symboles du signal. L'algorithme cherche la plus longue suite de symboles dans le dictionnaire. Puisque ce dernier contient des zéros, on pourra utiliser n'importe quelle chaîne de longueur 2. Dans cet exemple, on utilise la chaîne commençant à la position 9 (dernière position). Notons ici que la correspondance s'étend jusqu'à la fenêtre de lecture.

0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	2	1	0

1) C_1=(9,2,'1')

0	0	0	0	0	0	0	0	1	0	1	0	2	1	0	2	1	0

2) C_2=(8,3,'2')

0	0	0	0	1	0	1	0	2	1	0	2	1	0	2	1	2	4

3) C_3=(7,7,'2')

2	1	0	2	1	0	2	1	2	4	0	1	0	1	0	2	1	0

4) C_4=(0,0,'4')

1	0	2	1	0	2	1	2	4	0	1	0	1	0	2	1	0	0

5) C_5=(5,1,'1')

Travail demandé

Partie A

A.1. Programmer en MATLAB une fonction nommée lz77 permettant de compresser le signal de l'exemple ci-dessus (00101021021021240101021 00) en utilisant une fenêtre glissante de taille N uint8 et un dictionnaire de taille n1 uint8. Notez qu'il faut utiliser le type uint8 pour représenter les symboles du signal à coder. Chaque élément du triplet (position, longueur, suivant) sera également représenté par le type uint8.

A.2. Utilisez votre fonction dans un nouveau **script nommé « manipulation1.m »** pour compresser le signal pour chacune des Combinaisons de Paramètres (CP_i) suivantes :

CP_1 : N=4 et n1=2;

CP_2 : N=8 et n1=4;

CP_3 : N=16 et n1=8;

CP_4 : N=32 et n1=16.

A.3. Tracez la courbe du Taux de Compression (TC) en fonction des combinaisons de paramètres : TC= *ƒ* (CP).

A.4. Trouvez la meilleure combinaison de paramètres. Justifiez votre réponse.

Partie B

B.1. Écrivez un **script « manipulation2.m »** permettant de lire séquentiellement et de coder les pixels de l'image en niveaux de gris *'lena.gif'* [1] (disponible sur le site du cours, à la section des activités / TP1) en utilisant une fenêtre glissante de taille N uint8 et un dictionnaire de taille n1 uint8. Pour ce faire, vous devez utiliser la fonction lz77 implémentée précédemment.

B.2. Effectuez la compression de l'image avec les combinaisons de paramètres suivantes :

CP_1 : N=4 et n1=2;

CP_2 : N=8 et n1=4;

CP_3 : N=16 et n1=8;

CP_4 : N=32 et n1=16;

CP_5 : N=64 et n1=32;

CP_6 : N=128 et n1=64.

B.3. Tracez la courbe du Taux de Compression (TC) en fonction des combinaisons de paramètres : TC= *ƒ* (CP).

B.4. Trouvez la meilleure combinaison de paramètres. Commentez votre choix.

1. https://fr.wikipedia.org/wiki/Lenna

Partie C

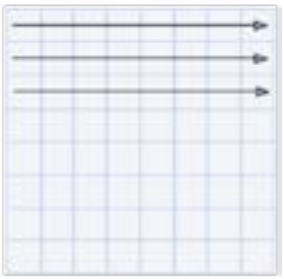
C.1. Considérez le fichier « manipulation1.m » et la fonction lz77 de la section (A). Dites s'il existe parmi les combinaisons de paramètres proposées, une combinaison permettant de traiter des chaînes de symboles dont la longueur peut atteindre les 300 symboles (permettant de coder par un triplet une chaîne dont la longueur atteint les 300 symboles). Justifiez votre réponse.

C.2. Expliquez tout changement éventuel à apporter à l'énoncé (A) afin de traiter des chaînes de symboles dont la longueur peut atteindre 300.

Indications :

L'image à coder doit être parcourue ligne par ligne et de gauche à droite, en commençant par le pixel le plus haut à gauche, comme cela est montré dans la figure ci-dessous.

Parcours d'une image à coder :



Les taux de compression doivent être calculés à l'aide de l'équation suivante :

Taux de Compr. = (Longueur du code en entrée / Longueur du code en sortie)

La fonction doit avoir l'entête *function code=lz77(signal, TailleFG,TailleDict)*

% *signal* : le signal à coder (entrée de l'algorithme).

% *TailleFG* : la taille de la fenêtre glissante.

% *TailleDict* : la taille de dictionnaire.

% *code* : un vecteur de type uint8 (résultat du codage lz77).