Rapport de TP - SY26

TP05 - La compression vidéo

Rémi Burtin

Cyril Fougeray

19 juin 2014

Université de Technologie de Compiègne

1 Introduction

Le but de ce TP est de mettre en œuvre la technique du *block matching* utilisée dans certains algorithmes de compression vidéo (codage inter-image, *P-Frame*).

Le principe est le suivant : l'image courante est divisée en blocs de taille MxM. On définit dans l'image de référence une zone de recherche de taille 2W+M placée autour de la position du centre du bloc de l'image courante. Chaque bloc dans la zone de recherche est ensuite comparé avec le bloc de l'image courante (on utilisera ici le critère *Mean Squared Difference*).

2 Mise en oeuvre du block matching

2.1 Padding de l'image

La division de l'image en bloc implique que la largeur et la hauteur de l'image soient respectivement multiples de la largeur et de la hauteur des blocs. Si ce n'est pas le cas, nous complétons avec des zéros (pixels noirs) grâce à la fonction **padarray** et son paramètre DIRECTION que nous fixons à 'post' afin d'ajouter les pixels après. Pour calculer le nombre de pixels à rajouter en largeur on utilise la formule suivante :

$$(M - (largeur(image) \mod M)) \mod M$$

avec M largeur d'un bloc $= 2^N + 1$, N un entier.
Dans le cadre de cet exercice nous faisons varier N entre 1 et 3.

NB. Afin de simplifier les calculs, nous convertissons les images en niveaux de gris afin de les manipuler plus aisément.

FIGURE 1 – Image, issue de la video *garden*, que l'on a voulu diviser en bloc de 7x7. L'image faisant 352x240, il a fallu rajouter 5 pixels noirs en largeur et 5 en hauteur.

2.2 Calcul de la fenêtre de recherche

Pour chaque bloc de l'image, nous devons maintenant calculer une fenêtre de recherche centrée sur le bloc courant et d'une taille de $2W+M$ (W prendra les valeurs 5, 10 et 15 dans notre cas). Afin de calculer les coordonnées de cette fenêtre pour chacun des blocs, nous parcourons tout d'abord tous ces blocs grâce à l'imbrication de deux boucles, ce qui nous permet de calculer les coordonnées du premier pixel dans le coin supérieur gauche du bloc puis de calculer la fenêtre de recherche via la fonction *search_window* (*cf.* A.2) :

```
% Pour i allant de 1 à hauteur de l'image, par pas de M.
for i=1:M:size(img,1),
% Pour i allant de 1 à largeur de l'image, par pas de M.
for j=1:M:size(img,2),
% extraction bloc courant
block = img(i:i+M-1,j:j+M-1);
%calcul de la fenetre de recherche
[window, orig_x, orig_y] = search_window(img_ref,W,M,i,j);
[...]
end;
end;
```

La fonction *search_window* renvoie trois paramètres :
- *window* : matrice constituée des pixels de la fenêtre de recherche
- *orig_x* : coordonnée en x du bloc courant dans la fenêtre de recherche
- *orig_y* : coordonnée en y du bloc courant dans la fenêtre de recherche

Afin de calculer les coordonnées dans l'image de la fenêtre de recherche, nous ajoutons (resp. soustrayons) W aux coordonnées horizontales et verticales des bords droit et bas (resp. gauche et haut) du bloc courant, tout en faisant attention à les coordonnées calculées ne débordent pas de l'image.

2.3 Recherche du meilleur bloc

Pour la recherche du bloc le plus similaire compris dans la fenêtre de recherche, nous avons créé une fonction prenant en paramètre le bloc, la fenêtre ainsi que les coordonnées du bloc dans la fenêtre (*cf.* § précédent).

Le but est de trouver le vecteur de mouvement telle que l'erreur moyenne, calculé via le critère MSD, soit minimale. Nous avons mis en place une méthode de recherche complète (dans toute la fenêtre) ce qui est une solution lente à exécuter mais simple à mettre en œuvre, comparé à la méthode de recherche logarithmique à deux dimensions ou à l'estimation de mouvement dite hiérarchique. Le but ici est donc de parcourir toute la fenêtre avec le bloc courant : nous implémentons cette étape grâce à deux boucles imbriquées afin de parcourir les lignes et les colonnes. Pour chaque position du bloc dans la fenêtre de recherche, nous calculons le MSD. Nous avons pour cela implémenté un fonction, *compute_msd* (*cf.* A.1). Cette fonction implémente la formule suivante (C étant le bloc dans l'image courante, P dans l'image prédite) :

$$MSD = \frac{1}{M^2} \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} (C(i,j) - P(i,j))^2$$

Nous sauvegardons le minimum de tous les MSD calculés ainsi que le vecteur de mouvement pour le bloc correspondant.

2.4 Reconstruction de l'image prédite

Nous construisons l'image prédite à partir des blocs de l'image de référence. Le bloc ij de l'image prédite correspondra au bloc de l'image de référence (que nous avons trouvé lors de l'étape précédente) qui correspond le mieux au bloc ij de l'image courante. Pour récupérer ces bloc, nous utilisons les vecteurs de déplacement calculés précédemment :

```
ref_block = img_ref(i+delta_y : i+M-1+delta_y ,j+delta_x : j+M-1+delta_x);
output_image(i : i+M-1 ,j : j+M-1) = ref_block;
```

3 Résultats

Pour montrer nos résultats, nous allons utiliser deux images issue de la séquence *garden* (très utilisée dans le domaine du traitement vidéo, comme Lena pour le traitement des images). Voici les deux images que nous allons utiliser :

FIGURE 2 – Frames 2 et 5 de la séquence garden

Le MSD entre ces deux images s'élève à 3465. Voici l'image résultante de la différence entre l'image courante (Frame 5) et l'image de référence (Frame 2) :

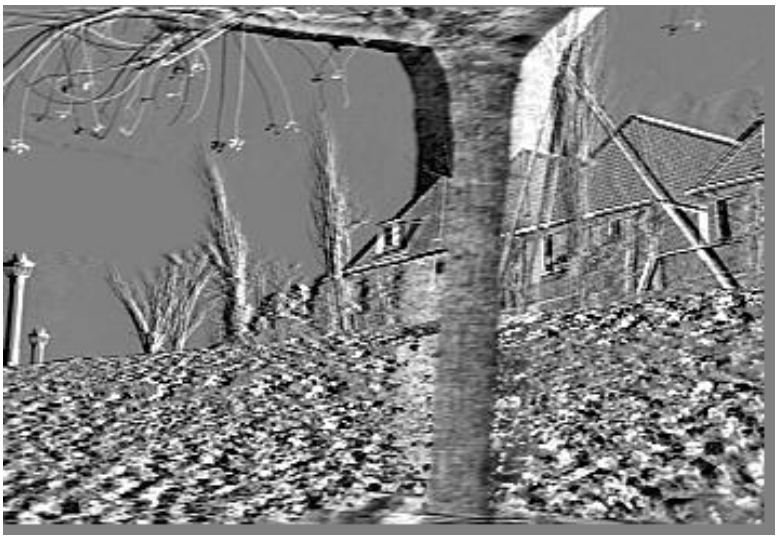

FIGURE 3 – Différence entre image courante et image de référence

On commence par un effectuer un block matching avec des blocs d'une taille 7x7 et une fenêtre de recherche de 17x17 (ou moins selon la position du bloc courant dans l'image) :

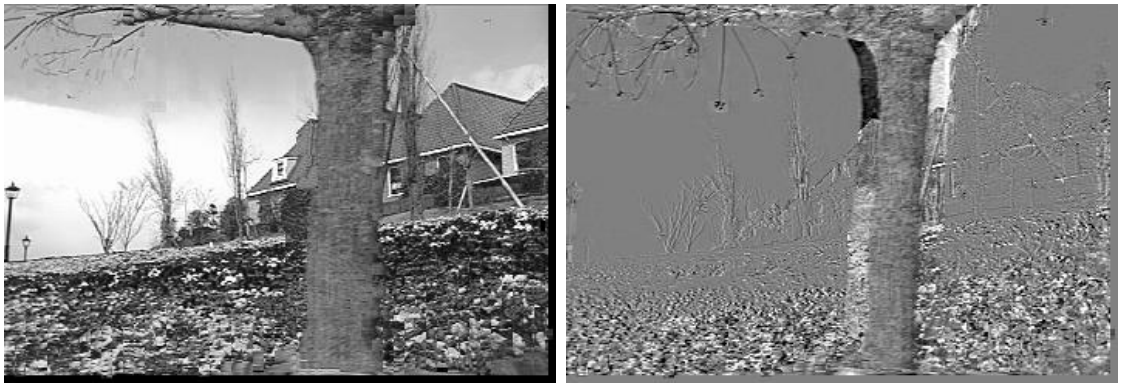

FIGURE 4 – Image courante prédite et erreur de prédiction pour $N = 3$ et $W = 5$. Le MSD est alors de 1128.

On passe ensuite à des blocs 3x3 et une fenêtre de recherche 33x33. On devra donc analyser beaucoup plus de positions mais la prédiction sera beaucoup plus fine. En effet on voit que le MSD tombe à 188 :

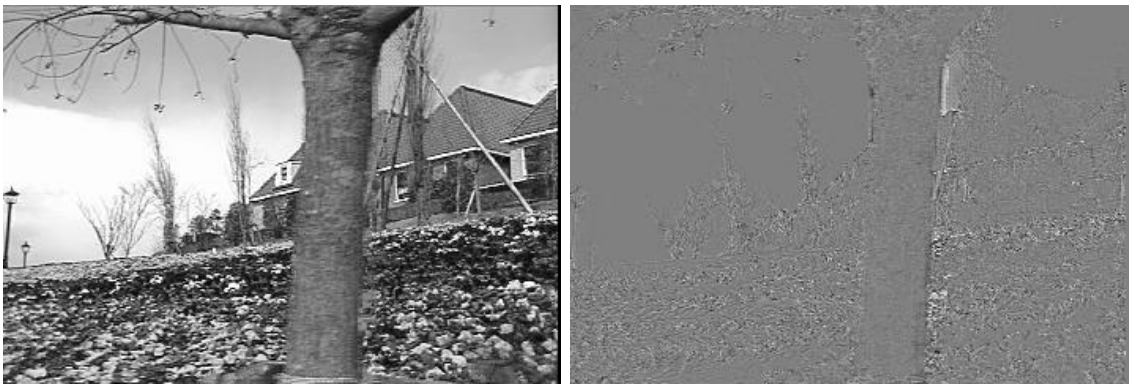

FIGURE 5 – Image courante prédite et erreur de prédiction pour $N = 1$ et $W = 15$. Le MSD tombe alors à 188.

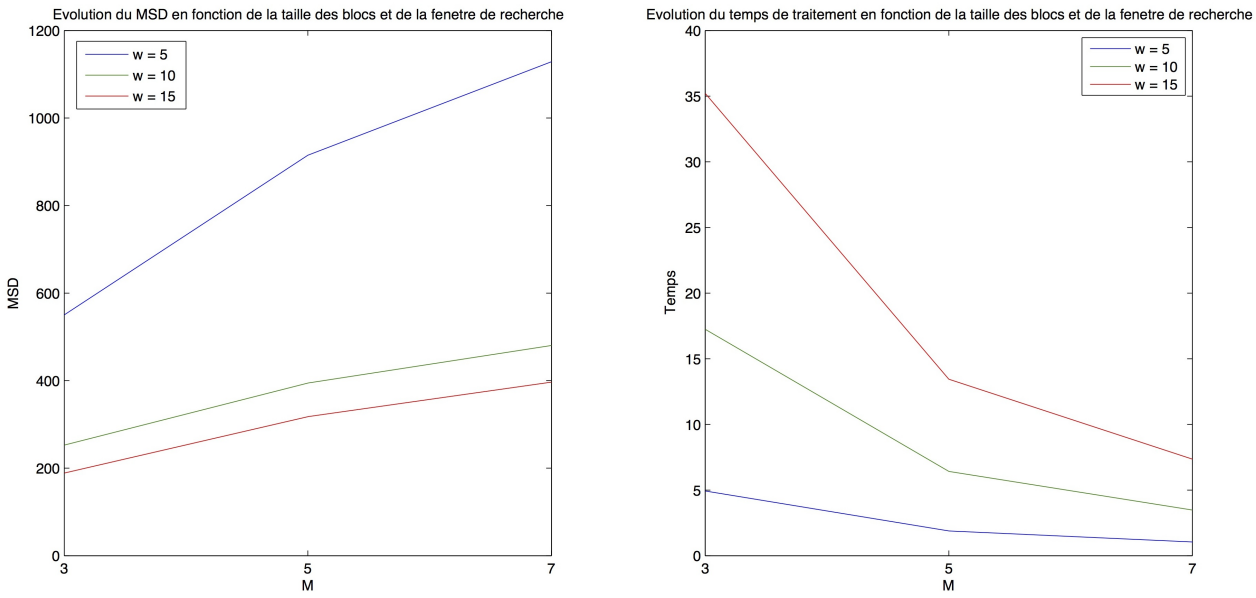

FIGURE 6 – Evolution du MSD et du temps de calcul en fonction de M et W

On observe que plus la fenêtre de recherche est grande, plus le MSD sera faible, mais plus le temps de calcul sera important. Et inversement pour la taille des blocs : plus les blocs seront grands, plus le MSD sera important et moins le temps de calcul sera important.

Il peut être aussi intéressant d'afficher l'ensemble des vecteurs de mouvement, pour observer le déplacement des différents bocs entre l'image courante et l'image de référence :

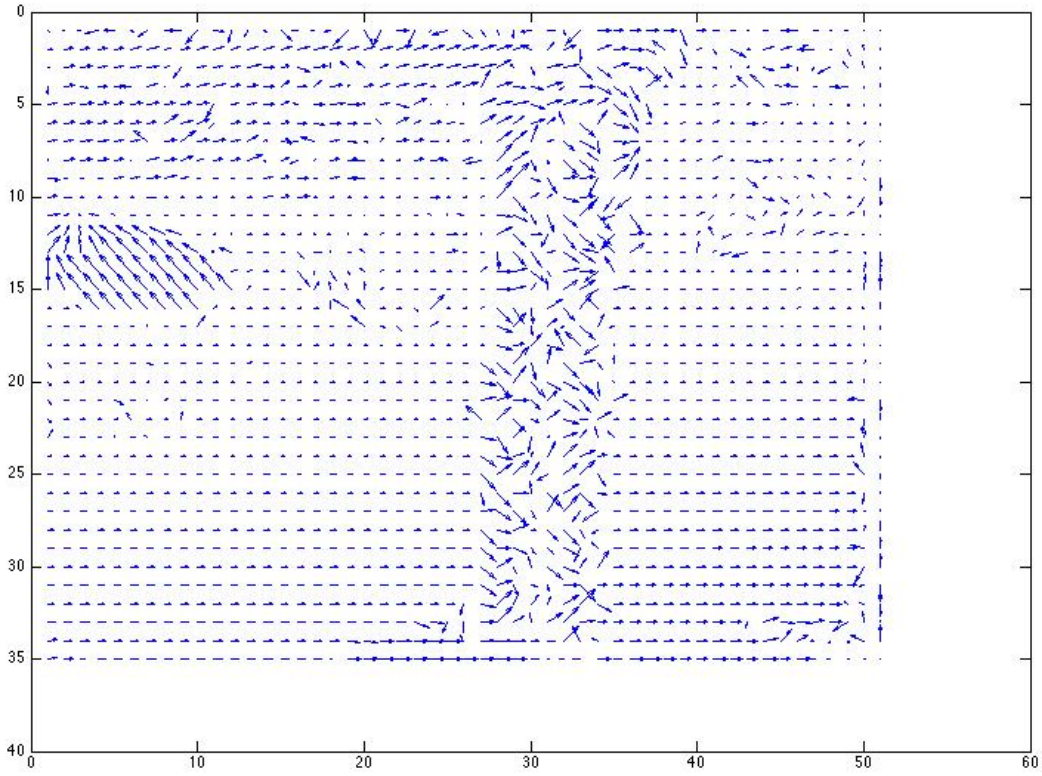

FIGURE 7 – Vecteurs de mouvement pour $N = 3$ et $W = 15$.

On voit que nous arrivons bien à compenser le travelling qui s'opère sur l'arrière plan. Mais que c'est un peu plus difficile sur l'arbre qui est au premier plan.

On génére ce champs de vecteur en sauvegardant chaque composante des vecteurs (x et y) dans deux matrices séparées. Puis on appelle la fonction *quiver* en lui passant la matrice comportant les composantes x et la matrice comportant les composantes y.

4 Conclusion

Ce TP nous a permis de comprendre par la pratique l'encodage prédictif inter-image (P-frame), utilisé dans la plupart des normes de compression vidéo (H261, H263, MPEG...) conjointement à d'autres techniques afin de réduire le taux d'erreur. Ainsi, le standard H261 inclut régulièrement dans la séquence des images encodées en suivant le principe de la norme JPEG (I-Frames). La norme MPEG ajoute aussi des frames bidirectionnelles. En pratique, les vecteurs de mouvement ainsi que les erreurs de prédiction sont encodés afin de supprimer des bits de redondance (en utilisant notamment une transformation par DCT, une quantification scalaire...) mais cette étape n'a pas été traitée lors de ce TP. Il serait aussi intéressant de mettre en œuvre l'algorithme de recherche logarithmique ou l'estimation de mouvement hiérarchique afin de retrouver plus rapidement le bloc le plus semblable dans la fenêtre de recherche.

A Codes source MATLAB

A.1 Calcul du MSD

```
function msd = compute_msd( current_block, block )
    M = size(block,1);
    N = size(block,2);
    msd = 1/(M*N);
    somme = 0;

    for i=1:M,
        for j=1:N,
            somme = somme + (current_block(i,j)-block(i,j))^2;
        end;
    end;
    msd = msd * somme;
end
```

A.2 Calcul de la fenêtre de recherche

```
function [ window, orig_x, orig_y ] = search_window( img, W, M, i, j )
    %i,j : indices dans l'image, du pixel en haut a gauche du bloc
    %M : largeur/hauteur du bloc
    %W : taille zone de recherche
    y1 = i-W;
    if y1 <= 0
        y1 = 1;
    end;
    y2 = i + (M-1) + W;
    if y2 > size(img,1)
        y2 = size(img,1);
    end;

    x1 = j-W;
    if x1 <= 0
        x1 = 1;
    end;
    x2 = j + (M-1) + W;
    if x2 > size(img,2)
        x2 = size(img,2);
    end;

    %indices du bloc courant dans la fenetre de recherche (utile pour le
    %calcul du vecteur de mouvement)
    orig_y = (i - y1)+1;
    orig_x = (j - x1)+1;

    %recuperation de la fenetre de recherche
    window = img(y1:y2,x1:x2);
end
```

A.3 Recherche d'un bloc dans l'image de référence

```
function [delta_x,delta_y,error] = block_matching(current_block,search_window,orig_i,orig_j)
    %initialisation du MSD minimum a Infini
    min_msd = inf;

    M = size(current_block,1);

    for i=1:(size(search_window,1)- M) + 1
        for j=1:(size(search_window,2)- M) +1
            %recuperation d'un bloc dans la fenetre de recherche
            block = search_window(i:(i+M)-1,j:(j+M)-1);

            %calcul du msd entre le bloc courant et le bloc que nous venons
            %de recuperer dans la fenetre de recherche
            msd = compute_msd(current_block, block);

            if msd < min_msd
                %sauvegarde du msd
                min_msd = msd;

                %calcul du vecteur de mouvement
                delta_y = i - orig_i;
                delta_x = j - orig_j;

                %calcul de l'erreur de prediction
                error = current_block - block;
            end;
        end;
    end;
end
```

A.4 Block Matching

```
function [ time, output_image, error, msd ] = block_matching_encode( img_ref, img, N, W )
    M = 2*N+1;

    %Conversion en niveaux de gris
    img_ref = rgb2gray(img_ref);
    img = rgb2gray(img);

    %padding de l'image (on ajoute des 0 pour que la largeur/hauteur soit
    %multiple de M
    pad_y = M - mod(mod(size(img,1),M), M);

    pad_x = M - mod(mod(size(img,2),M), M);

    img = double(padarray(img,[pad_y pad_x],0,'post'));
    img_ref = double(padarray(img_ref,[pad_y pad_x],0,'post'));

    %affichage de la difference entre l'image courante et l'image de
    %reference
    figure('name','Diff current vs ref');
    imshow(((img-img_ref)+128)/255);

    %Initialisation matrice contenant les composantes x et y des vecteurs
    %de mouvement
    matx = zeros(size(img,1)/M, size(img,2)/M);
    maty = zeros(size(img,1)/M, size(img,2)/M);

    %timer
    tic

    k=1;

    for i=1:M:size(img,1), % for i=1 to i=height, with a step size of M.
        l=1;
        for j=1:M:size(img,2),
            %extraction bloc courant
            block = img(i:i+M-1,j:j+M-1);

            %calcul de la fenetre de recherche
            [window, orig_x, orig_y] = search_window(img_ref,W,M,i,j);

            %recherche du bloc qui se rapproche le plus du bloc courant
            %dans la fenetre de recherche
            [delta_x, delta_y, error] = block_matching(block,window,orig_y,orig_x);

            %on met le vecteur de mouvement dans des matrices (une pour la
            %composante x et une autre pour la y)
            matx(k,l) = delta_x;
            maty(k,l) = delta_y;

            l = l + 1;

            %on recupere le bloc que nous avons trouve dans l'image de ref
            %grace au vecteur de mouvement et le mettons a la place qu'il
```

```
54              %devrait avoir dans l'image courante
55              ref_block = img_ref(i+delta_y : i+M-1+delta_y , j+delta_x : j+M-1+delta_x);
56              output_image(i:(i+M)-1,j:(j+M)-1) = ref_block;
57
58          end;
59          k = k + 1;
60      end;
61      time = toc;
62
63      %calcul et affichage de l'erreur de prediction
64      error = (img - output_image)+128;
65      figure('name',strcat('Erreur de prediction n=',num2str(N),' w=', num2str(W)));
66      imshow(error/255);
67
68      msd = compute_msd(img, output_image);
69
70      %champ de vecteur
71      figure('name',strcat('Vecteurs de mouvement n=',num2str(N),' w=', num2str(W)));
72      quiver(matx/M,maty/M);
73      %inversion de l'axe des y
74      axis ij;
75
76
77      %affichage de l'image predite
78      figure('name',strcat('Image predite n=',num2str(N),' w=', num2str(W)));
79      imshow(output_image/255);
80  end
```

A.5 Script de test du Block Matching

```
W = 15;
N = 3;

i1 = imread('garden1.jpg');
i2 = imread('garden2.jpg');

%i1 = imread('man1.jpg');
%i2 = imread('man2.jpg');

%i1 = imread('car1.jpg');
%i2 = imread('car2.jpg');

%i1 = imread('girl1.jpg');
%i2 = imread('girl2.jpg');

[time, output_image, error, msd] = block_matching_encode(i1,i2,N,W)
```