Rapport de TP - SY26

TP05 - La compression vidéo

Rémi Burtin

Cyril Fougeray

17 juin 2014

Université de Technologie de Compiègne

1 Introduction

Le but de ce TP est de mettre en oeuvre la technique du block matching utilisée dans certains algorithmes de compression vidéo.

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. 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.

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

2.3 Recherche du meilleur bloc

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) :

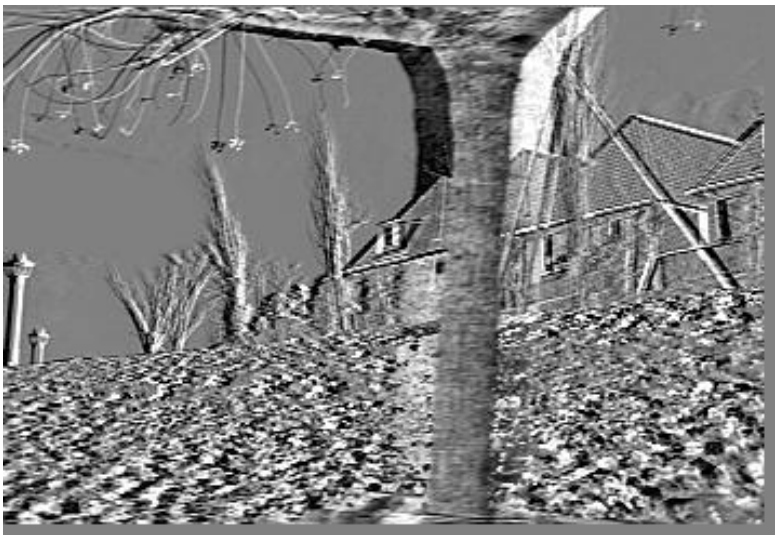

FIGURE 3 – Différence entre image courante et image de référence

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) :

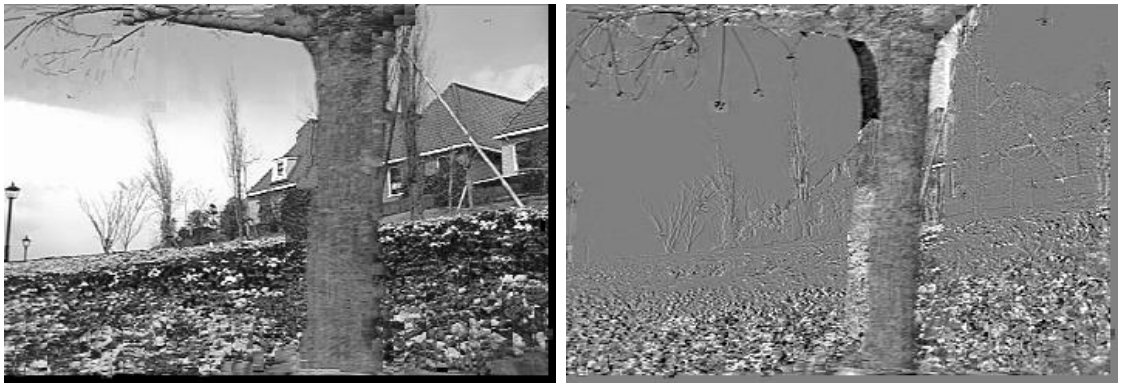

FIGURE 4 – Image courante prédite et erreur de prédiction pour $N = 3$ et $W = 5$. Le MSD est alors de 1128.

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 :

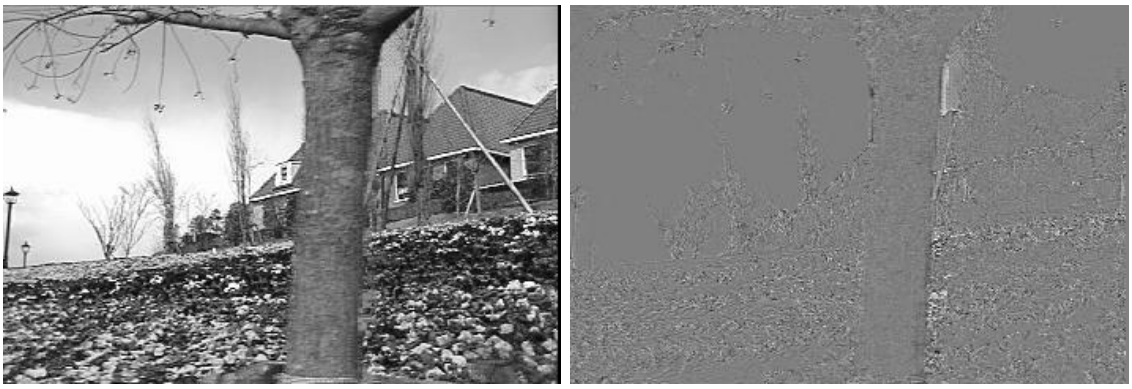

FIGURE 5 – Image courante prédite et erreur de prédiction pour $N = 1$ et $W = 15$. Le MSD tombe alors à 188.

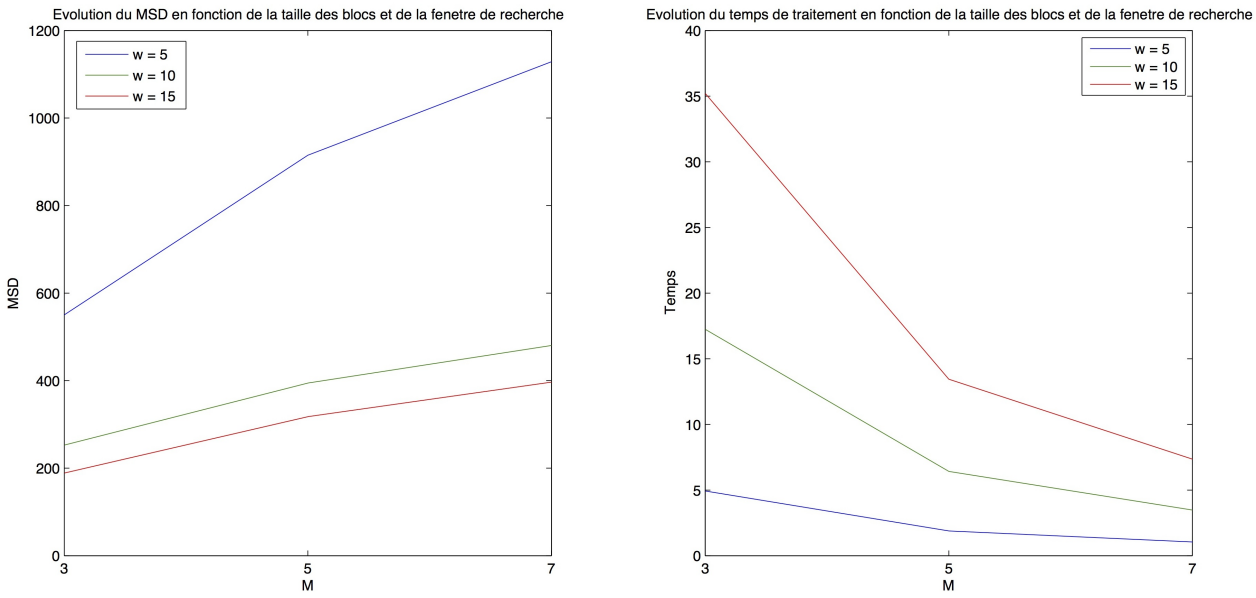

FIGURE 6 – Evolution du MSD et du temps de calcul en fonction de M et W

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 calul sera important.

4 Conclusion

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 = mod(size(img,1),M);
    if pad_y > 0
        pad_y = M - pad_y;
    end;

    pad_x = mod(size(img,2),M);
    if pad_x > 0
        pad_x = M - pad_x;
    end;
    imwrite(img,'garden.jpg','jpg');
    img = double(padarray(img,[pad_y pad_x],0,'post'));
    imwrite(img/255,'garden_padding.jpg','jpg');
    img_ref = double(padarray(img_ref,[pad_y pad_x],0,'post'));

    %affichage de la difference en tre 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),
        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;
```

```
54              maty(k,l) = delta_y;
55
56              l = l + 1;
57
58              %on recupere le bloc que nous avons trouve dans l'image de ref
59              %grace au vecteur de mouvement et le mettons a la place qu'il
60              %devrait avoir dans l'image courante
61              ref_block = img_ref(i+delta_y : i+M-1+delta_y , j+delta_x : j+M-1+delta_x);
62              output_image(i:(i+M)-1,j:(j+M)-1) = ref_block;
63
64          end;
65          k = k + 1;
66      end;
67      time = toc;
68
69      %calcul et affichage de l'erreur de prediction
70      error = (img - output_image)+128;
71      figure('name',strcat('Erreur de prediction n=',num2str(N),' w=', num2str(W)));
72      imwrite(error/255,strcat('garden_error_n_',num2str(N),'_w_',num2str(W),'.jpg'),'jpg'););
73
74      msd = compute_msd(img, output_image);
75
76      %champ de vecteur
77      figure('name',strcat('Vecteurs de mouvement n=',num2str(N),' w=', num2str(W)));
78      quiver(matx/M,maty/M);
79      %inversion de l'axe des y
80      axis ij;
81
82
83      %affichage de l'image predite
84      figure('name',strcat('Image predite n=',num2str(N),' w=', num2str(W)));
85      imwrite(output_image/255,strcat('garden_pred_n_',num2str(N),'_w_',num2str(W),'.jpg'),'jpg'
86  end
```