Institut Galilée
Télécom 3

Travaux pratiques de traitement d'images numériques
Troisième séance
Institut Galilée
2010-2011

G.Dauphin et A. Beghdadi

L'estimation de mouvement consiste à déterminer les vecteurs de mouvements qui permettent de prédire la transformation d'une image à une autre image dans une séquence vidéo. Les vecteurs de mouvements peuvent concerner l'ensemble de l'image ou une partie de l'image, par exemple des blocs rectangulaires. Ces vecteurs de mouvement peuvent correspondre à une simple translation ou plus généralement à n'importe quel modèle capable d'approcher le mouvement d'une caméra réelle ou le mouvement d'objets dans la scène.

Appliquer les vecteurs de mouvement à une image de façon à synthétiser une approximation de l'image suivante, c'est faire de la compensation de mouvement. La combinaison de l'estimation de mouvement et de la compensation de mouvement est un élément important de la compression vidéo faite par MPEG 1,2 et 4.

On cherche tout d'abord à construire une image ayant des blocs disparus. Les fonctions suivantes permettent de supprimer une colonne et de rajouter une colonne :

```
>>supprimeCol=inline('[im(:,[1:end~=colEnleve]) im(:,end)]','im','colEnleve');
>>rajoutCol=inline('im(:,sort([1:end-1 colRajoute]))','im','colRajoute');
```

Leur fonctionnement peut être illustré en les appliquant à une matrice simple

```
>>ex1=reshape(1 :20,5,4).', supprimeCol(ex1,2), rajoutCol(ex1,2),
```

Grâce à une double application de la commande transpose (.'), on peut utiliser ces fonctions pour enlever et rajouter des lignes.

```
>>ex1=reshape(1 :20,5,4).' , supprimeCol(ex1.',2).', rajoutCol(ex1.',2).',
```

Les commandes suivantes permettent de choisir aléatoirement un numéro de ligne et un numéro de colonne.

```
>>lig= ceil(size(imMod,1)*rand(1));
>>col= ceil(size(imMod,2)*rand(1));
```

> Utilisez ces commandes sur une image en niveaux de gris notée im de façon à supprimer deux lignes et deux colonnes puis à rajouter deux autres lignes et deux autres colonnes. Cette nouvelle image est notée imMod. Comparez le rendu visuel de imMod et de im. Expliquez pourquoi il est difficile de détecter les déplacements de blocs.

La commande suivante permet de rendre visibles ces déplacements de blocs :

>>figure(1); imshow(0.5+0.5*abs(im-imMod));

On se donne une distance pour mesurer la distortion. Ici il s'agit

$$d(f_{ij}^o, f_{ij}^d) = \sum_{ij} \left| f_{ij}^o - f_{ij}^d \right|$$

où f_{ij}^o et f_{ij}^d sont les composantes de deux images ou de deux blocs de mêmes tailles. Cette distance est parfois aussi appelée distance induite par la norme L1.

Cette distance peut être implémentée par la commande suivante

>>diBlk=inline('sum(sum(abs(blk1-blk2)))','blk1','blk2');

On peut remarquer que cette norme est beaucoup plus sensible au déplacement de blocs que ne l'est l'oeil humain. Expliquez pourquoi les commandes suivantes permettent de l'affirmer.

>>diBlk(im,im+0.05*randn(size(im))),
>>diBlk(im,imMod),

On cherche maintenant à apparier un bloc de imMod avec un bloc de im. On considère ici des blocs de tailles 16x16. On suppose qu'il y a un bloc similaire à celui de imMod dans im et ce relativement près, en l'occurrence avec un déplacement inférieur ou égal à 7 pixels (en haut, en bas, à gauche ou à droite). On note ici les coordonnées du bloc de imMod : (blk1,blk2) et (blk1,blk2+15), (blk1+15,blk2+15), (blk1+15,blk2) pour les trois autres extrémités du bloc. Afin d'éviter les débordements, ces coordonnées doivent vérifier les conditions suivantes :

$$\begin{aligned} 7 \le blk_1 \le nb_{lig} - 1 - 7 - 15 \\ 7 \le blk_2 \le nb_{col} - 1 - 7 - 15 \end{aligned} \quad (1)$$

où $nb_{lig} \times nb_{col}$ est la taille de l'image. Pour s'adapter aux conventions de Matlab, il faut rajouter 1 à blk1 et blk2 dans ces deux inégalités. La commande suivante parcourt tous les déplacements inférieurs ou égaux à 7 pixels et pour chaque déplacement évalue la distance entre le bloc de imMod et le bloc de im associé à chaque déplacement. Notez qu'il n'y a pas d'apostrophes sur *bkd* dans la troisième instruction.

>>bkd=imMod(8:8+15, 8:8+15);
>>bko=im(1:8+15+7, 1:8+15+7);
>>diMatLoc=nlfilter(bko,[16 16],'sum(sum(abs(x-P1)))',bkd);

On aurait pu s'attendre à ce que diMatLoc soit de taille 15x15 du fait des déplacements possibles en réalité Matlab rajoute d'autres positions possibles en débordant au-delà de la zone délimitée. diMatLoc(15,15) indique la distortion s'il n'y a aucun déplacement.

On cherche quel est le déplacement pour laquelle la distortion est minimale. Le vecteur mouvement est obtenu en cherchant la coordonnée de la composante minimale de diMatLoc. L'implémentation proposée ici transforme diMatLoc en vecteur colonne, trouve la coordonnée

de la composante minimale, id et retrouve à partir de cette coordonnée les coordonnées de la composante minimale au sein de la matrice et finalement en déduit le vecteur de mouvement sachant qu'en la coordonnée 15,15 il n'y a pas de déplacement.

```
>>[Mn,MnId]=min(diMatLoc(:));
>>[MnId1,MnId2]=ind2sub(size(diMatLoc),Mnid);
>>mvt1=MnId1-15; mvt2=MnId2-15;
```

On cherche maintenant à appliquer cela sur toute l'image. On cherche les coordonnées de tous les blocs qui vérifient les conditions (1).

```
>>[id2,id1]=meshgrid(1:size(im,2),1:size(im,1));
>>bkId1=id1(8:16:end-7-15,8:16:end-7-15);
>>bkId2=id2(8:16:end-7-15,8:16:end-7-15);
```

Il suffit alors pour trouver le vecteur de mouvement pour chaque bloc de faire une boucle sur chaque bloc ainsi déterminé.

```
>>mvtP1=zeros(size(bkId1));
>>mvtP2=zeros(size(bkId1));
>>for k=1:length(bkId1(:))
>>  bko=im(bkId1(k)-7:bkId1(k)+15+7,bkId2(k)-7:bkId2(k)+15+7);
>>  bkd=imMod(bkId1(k):bkId1(k)+15,bkId2(k):bkId2(k)+15);
>>  diMatLoc=nlfilter(bko,[16 16],'sum(sum(abs(x-P1)))',bkd);
>>  [Mn,MnId]=min(diMatLoc(:));
>>  [MnId1,MnId2]=ind2sub(size(diMatLoc),MnId);
>>  mvtP1(k)=min(7,max(-7,MnId1-15));
>>  mvtP2(k)=min(7,max(-7,MnId2-15));
>>end;
```

> Comparez les matrices mvtP1 et mvtP2 en fonction des lignes et des colonnes que vous avez choisies de supprimer ou de rajouter.

On cherche enfin à calculer imDiff la différence entre imMod et im modifiée par la compensation de mouvement issue des matrices mvtP1 et mvtP2.

```
>>imDiff=zeros(size(im));
>>for k=1:length(bkId1(:))
>>  ids1=bkId1(k):bkId1(k)+15; ids2=bkId2(k):bkId2(k)+15;
>>  imDiff(ids1,ids2)=imMod(ids1,ids2)-im(ids1+mvtP1(k),ids2+mvtP2(k));
>>end;
```

> Comparez imDiff et im-imMod, pour visualisez de telles images qui ne sont pas à valeurs dans [0,1] mais dans [-1,1], il est préférable de multiplier par 0.5 puis de rajouter 0.5 avant la visualisation par imshow. Pourquoi la comparaison de ces images justifie l'intérêt de la détection de mouvement dans la compression vidéo.

Le fichier daysailerGray.zip contient 120 images jpeg au format qcif (176x144) en niveaux de gris.

Appliquez la détection de mouvement puis la compensation de mouvement à ces images et utilisez la compression jpeg implémentée dans imwrite pour stocker les différentes images imDiff. Contrôlez le rendu visuel de la vidéo après compression. Comparez les tailles des fichiers nécessaires pour stocker les deux vidéos.

Voici les instructions pour lire et visualiser les images en niveaux de gris dans dailysailerGray.zip

```
>>k=1;
>>for num=2764:2883
>>if num<9
>>  nom=['daysailer-00' num2str(num) '.jpeg'];
>>else
>>  if num<99 nom=['daysailer-0' num2str(num) '.jpeg'];
>>  else nom=['daysailer-' num2str(num) '.jpeg'];
>>  end;
>>end;
>>im=imread(['RGB' nom]);
>>im1=zeros([size(im) 3]); im1(:,:,1)=im; im1(:,:,2)=im; im1(:,:,3)=im;
>>M(k)=im2frame(im1); k=k+1;
>>end;
>>figure(1); movie(M);
```

Le fichier daysailerRGB.zip contient la même vidéo en format couleur. Pour faire la détection de mouvement dans une vidéo couleur, on cherche l'appariement entre blocs indépendamment de la composante couleur. C'est-à-dire que diMatLoc a la même taille que précédemment, mais la distance entre blocs est maintenant calculée en ajoutant la précédente distance entre blocs pour la composantes rouge, puis la composante verte et enfin pour la composante bleue. La compensation de mouvement se fait composante couleur par composante couleur. C'est-à-dire que imDiff est obtenue pour chaque bloc par la différence entre imMod pour ce bloc avec le bloc déplacé de im.

Répondez aux mêmes questions pour cette vidéo. Comparez aussi les détections de mouvement obtenues avec le format couleur et avec le format en niveaux de gris.