Ecole Nationale Supérieure de Cognitique (Bordeaux INP) – 2A

Remplacement de contenu

Traitement du signal

Grégoire MELIN – FLORIAN CAILLEAU

12/12/2016

TABLE DES MATIERES

INTRODUCTION

CONCEPT DE REALITE AUGMENTEE

La réalité augmentée est l'ajout d'informations virtuelles (dans le sens où elle n'existe pas naturellement) à la réalité en temps réel. Ces applications sont nombreuses. Parmi elles, on peut citer l'ajout de contenu publicitaire dans les retransmissions sportives, la réalisation de studios virtuels dans le cinéma ou encore la visualisation de patrimoine historique disparu via les smartphones. La réalité augmentée s'applique aussi bien à la perception visuelle (superposition d'image virtuelle aux images réelles) qu'aux perceptions tactiles ou auditives par exemple.

POSITIONNEMENT DU PROJET

REMPLACEMENT DE CONTENU D'UNE VIDEO

Le projet traite des perceptions visuelles. En effet, le but du projet est l'ajout de contenu dans une vidéo ou en d'autres termes la superposition d'image virtuelle (dans le sens où elle n'existe pas originellement dans la vidéo) aux images « réelles » (issues de la vidéo). L'objectif de ce projet est d'incruster du contenu dans une vidéo. La vidéo met en scène une feuille blanche portant des petits cercles bleus dans les coins qu'on appellera picots. Le but est d'incruster du contenu dans le rectangle blanc par une segmentation colorimétrique.

Vidéo

Longueur	00:00:09
Largeur de trame	576
Hauteur de trame	520
Débit de données	309 Kbits/s
Débit total (en bits)	309 Kbits/s
Fréquence d'images	25 trames/s

CHOIX DE LA TECHNIQUE

Le choix de mettre sur cette feuille des picots n'est pas anodin. En effet, on peut distinguer deux approches de segmentation de la feuille dans la vidéo à savoir une détection à partir des contours de la feuille ou une détection à partir des picots. On optera dans ce projet pour une caractérisation de la zone de la feuille par rapport aux picots.

DEMARCHE ALGORITHMIQUE

DESCRIPTION GLOBALE

La démarche du projet est de détecter les picots par rapport à leur profil colorimétrique, les traquer toute la vidéo et à partir des informations sur ces picots d'incruster un contenu de son choix dans la zone déduite de l'analyse des picots sachant que cette dernière évolue en termes de formes et de position. En outre, certains éléments tels que les doigts ne permettent pas de voir la zone complète décrite par la feuille.

SPECIFICATION

GENERATION DU MODELE COLORIMETRIQUE DES PICOTS BLEUS : PREMIERE IMAGE

PARAMETRAGE DU MODELE

La première étape consiste à sélectionner l'une des images du modèle par exemple la première image de la vidéo. A partir de cette image, on sélectionne ensuite un picot bleu dans cette image. La lumière de la vidéo ne variant pas en termes d'intensité et de composition, on peut supposer que la couleur que l'on perçoit du picot est relativement stable.

DEFINITION DU MODELE

A partir de cet échantillon, on peut définir le modèle colorimétrique du picot. Le but est donc de savoir qu'est ce qui caractérise d'un point de vue colorimétrique le picot. On calcule pour cela la moyenne colorimétrique du picot. L'image est en vraie couleurs, permettant ainsi de décomposer l'image en trois composantes (Rouge, Vert, Bleue) et de calculer la moyenne de chacune de ses composantes. Pour déterminer le profil des picots on pourra caractériser leur profil colorimétrique par les distances de Mahalanobis. En calculant la distance de Mahalanobis pour un point de l'image on pourra quantifier la similarité (en termes de couleur) entre un point d'une image de la vidéo et le profil colorimétrique des couleurs calculé. Pour cela, on a besoin entre autre de calculer la moyenne colorimétrique des pixels composant le picot et la matrice de covariance associée.

SEUILLAGE

Le calcul de la distance de Mahalanobis des picots permet de segmenter les picots dans chaque image de la vidéo par seuillage des images de la vidéo. Le but est de déterminer si un pixel est assez similaire à ceux formant le picot (en utilisant sa distance de Mahalanobis) pour être considéré comme un pixel appartenant effectivement à un des quatre picots. On déterminera le meilleur seuil par essai-erreur aboutissant ainsi aux positions exactes des picots au sein de chaque image de la vidéo. On pourra ainsi définir un modèle d'un picot bleu définit par un seuil.

IDENTIFICATION & TRAQUAGE DES PICOTS SUR LA VIDEO : TRAITEMENT DES IMAGES SUIVANTES

ANALYSE

La première étape de segmentation des picots est de déterminer où se trouve les picots. Grâce à un seuillage sur la distance de Mahalanobis, on peut ainsi délimiter plusieurs zones correspondant aux quatre picots.

LABELLISATION & ORDONNANCEMENT DES PICOTS

Une fois les picots segmentés, il est nécessaire en premier lieu de labelliser les picots afin de les discriminer entre eux car, pour des raisons notamment de stabilité et de cohérence de l'image incrustée par rapport à la vidéo (elle ne doit pas se retourner subitement si la feuille dans laquelle elle est incrustée ne se retourne pas), il est nécessaire de pouvoir différencier le picot 1 (initialement en haut à gauche) et le picot 2 (initialement en haut à droite) par exemple.

LABELLISATION

On choisit pour labelliser les picots de les détecter et d'en faire une image seuillée (soit une image binaire) dont toutes les zones des « picots » sont représentées par des 1 (le reste de l'image étant représentée par des 0). On attributs à chaque zone un nombre en changeant la valeur 1 initial en 1, 2, 3, 4 de manière aléatoire.

IMPLEMENTATION

IMPLEMENTATION GLOBALE

GENERATION DU MODELES DES PICOTS & DES DOIGTS

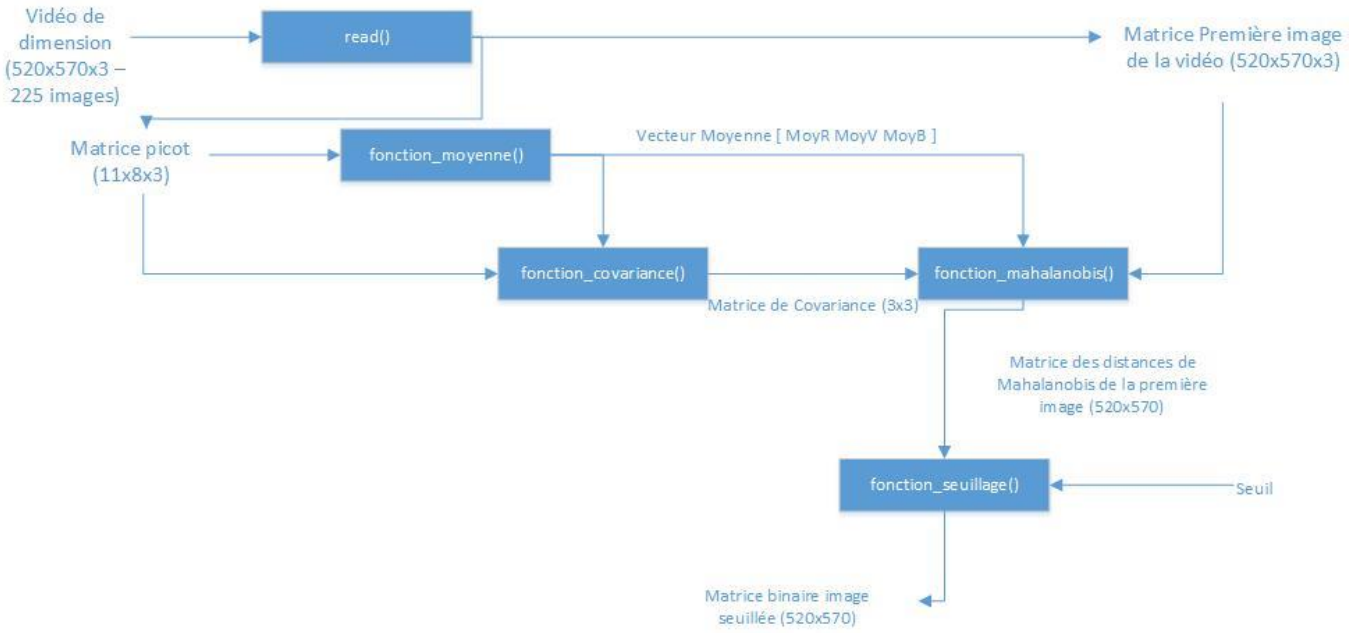


Fig. : Schéma-bloc de l'algorithme implémenté utilisé pour calculer le modèle colorimétrique des picots

La première étape de l'algorithme pour le calcul du modèle des picots est la sélection d'un d'entre eux. On utilise la fonction *VideoReader*() afin de transformer la vidéo en un objet lisible par Matlab. Suite à ça on sélectionne la première image. On sélectionne la zone de l'image où se situe un des picots.

Il est nécessaire maintenant de calculer les différentes grandeurs citées ci-dessus afin de pouvoir quantifier les caractéristiques colorimétriques des picots. Une première grandeur importante est la moyenne. Le choix a été fait d'écrire une fonction *fonction_moyennage.m* qui prend en entrée une image et renvoie un vecteur ligne de dimension trois où les trois éléments sont les moyennes de l'image des différentes composantes de la couleur. Cette fonction prend chacune des trois matrices composant l'image du picot et calcule la moyenne de la valeur des pixels de cette matrice.

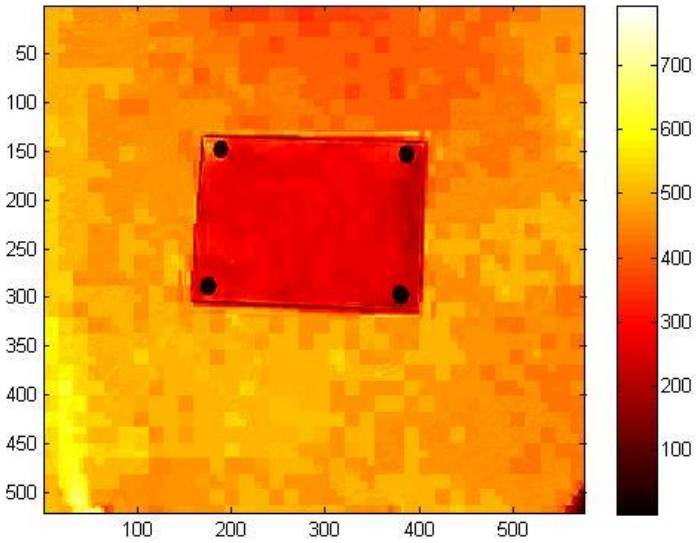
Il s'agit maintenant de calculer une deuxième grandeur statistique. On veut maintenant calculer la dispersion des «bleus » composant le picot. Le choix a été fait d'écrire une fonction *fonction_covariance.m* qui calcule la matrice de covariance du picot par le calcul de chaque coefficient à l'aide de la moyenne calculé précédemment. Ces fonctions n'ont pas grand-chose de remarquable dans le sens où elles calculent simplement les grandeurs statistiques moyenne et matrice de covariance.

Ces grandeurs statistiques sont déterminées dans le but de calculer les distances de Mahalanobis des pixels des images de la vidéo. Si on implémente directement la définition des distances de Mahalanobis, l'algorithme fonctionne. Cependant, on se rend vite compte qu'étant donné le nombre d'image, 30 secondes

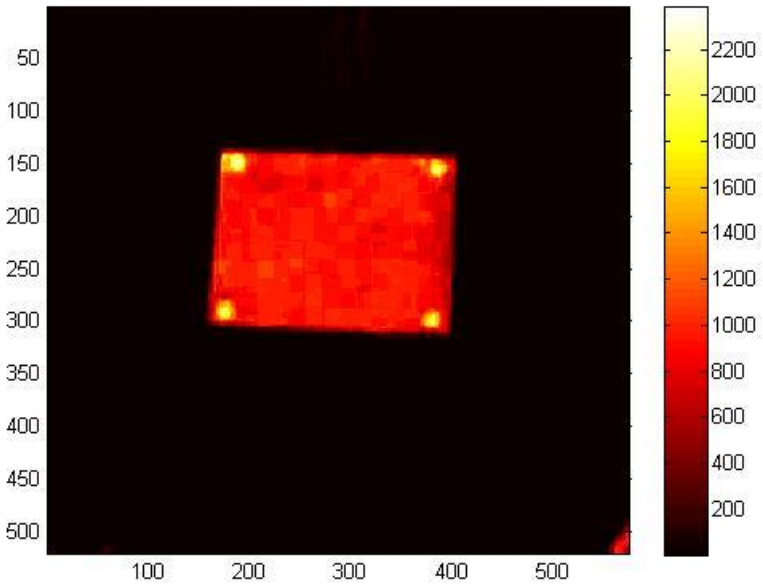
par image n'est pas d'un point de vue efficacité de l'algorithme très pertinent. C'est pour cela que suite à ce constat, le choix a été fait de calculer les distances de Mahalanobis qu'on appellera « simplifiées » dans le sens où l'on ne calcule pas toutes les distances de Mahalanobis. Mais on reviendra sur ce point par la suite.

Rem. : Le modèle des picots étant généré, on génère le modèle des doigts. Ce modèle est généré de manière analogue à celui des picots c'est pour cela qu'on ne rentrera pas dans le détail.

Ci-dessous, on représente les distances de Mahalanobis de la première image par rapport au modèle colorimétrique des picots en fausses couleurs. Dans l'image ci-dessous, on remarque que les distances de Mahalanobis au niveau des picots sont moins importantes au niveau des picots montrant ainsi que les pixels de ces zones sont similaires aux pixels du modèle définie plus haut.

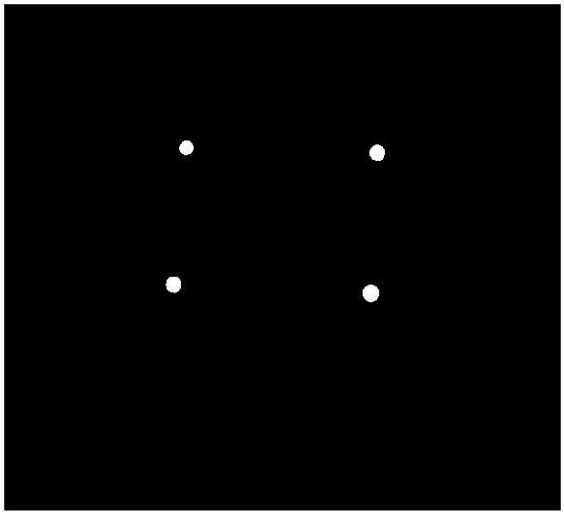


Ci-dessous, on peut observer la représentation de la première image en fausses couleurs mais cette fois ci par rapport au modèle des doigts :



Suite à ces opérations, il est nécessaire de pouvoir discriminer les picots. Pour cela, on utilise une technique de seuillage à travers la fonction de seuillage *fonction_seuillage* où l'on seuille l'image par rapport à la distance de Mahalanobis. On choisit alors le meilleur seuil permettant de sélectionner les 4 picots.

Le résultat du seuillage des picots est représenté à gauche et celui des doigts à droite.

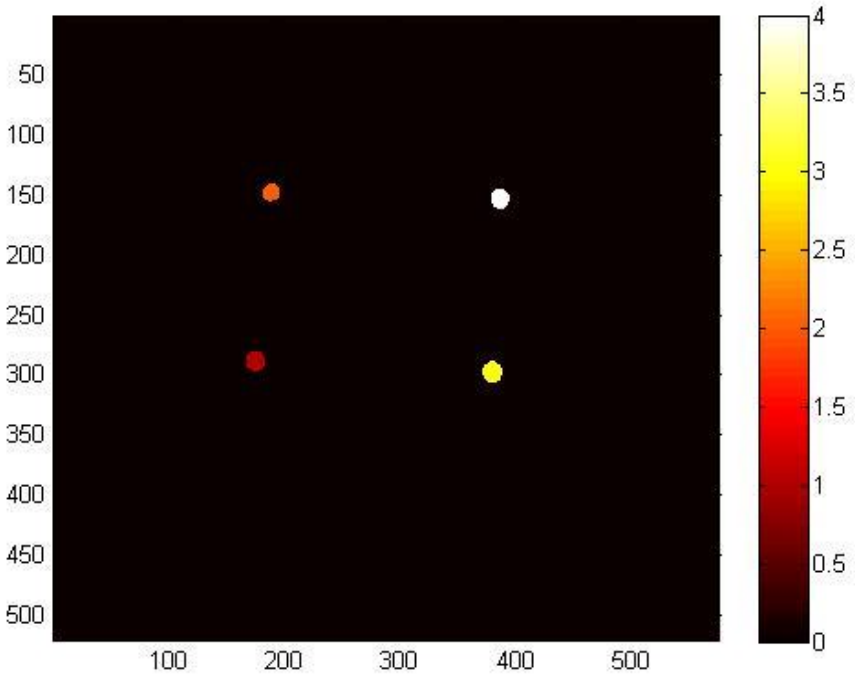


On sauvegarde ensuite les modèles obtenus grâce à la fonction *save.* On sauvegarde alors les objets suivants : le meilleur seuil obtenu pour sélectionner les picots, la matrice de covariance et le vecteur moyenne du picot ainsi que la distance de Mahalanobis de la première image de la vidéo. On sauvegarde également les données relatives aux doigts (meilleur seuil, moyenne, matrice de covariance, matrice des distances de Mahalanobis).

DETECTION, TRAQUAGE DES PICOTS

On seuille la première image. On calcule les barycentres des picots et on les ordonne grâce aux fonctions *fonction_labellisation* et la *fonction_ordre_picot.* On incruste ensuite le contenu dans la première image.

La première image nécessite un traitement plus important que sur le reste des images puisque l'enjeu est de repérer les picots d'une part mais également de les ordonner par la suite. La première étape de labellisation montre une image composée de 4 picots. On labellise les picots et ainsi chaque picot est associé à un entier. Voici la représentation en fausses couleurs de cette opération :



Reste à ordonner les picots puisque comme on peut l'observer sur la représentation ci-dessus, les valeurs n'ont pas de logique si on lit les valeurs des picots dans le sens horaire ou dans le sens antihoraire.

INCRUSTATION

Grâce à la fonction *fonction_incrustation.m,* on incruste du contenu dans la vidéo. Ici, on change le contenu incrusté selon le moment de la vidéo auquel on se trouve. Plus on est loin dans la vidéo, plus le personnage Sangoku apparaît sous des transformations puissantes.

Fig. : Prise de vue de la vidéo en début et fin de vidéo

FOCUS SUR LES FONCTIONS

FONCTION_MAHALANOBIS

Cette fonction applique le calcul de la distance de Mahalanobis à chaque pixel d'une image donnée. Pour réduire le temps de calcul de l'ordinateur nous l'avons simplifiée afin de recouper les matrices obtenues par le calcul classique et de traiter seulement les valeurs qui nous intéressent (fonction Reshape).

FONCTION_SEUILLAGE

Cette fonction compare chaque pixel de l'image (modifiée avec la distance de Mahalanobis) à la valeur de seuil déterminée. Si la valeur du pixel est supérieure alors on lui donne la valeur 0 et inversement.

FONCTION_INCRUSTATION

Cette fonction permet l'insertion d'une image dans la vidéo. Pour cela on fournit l'image que l'on veut incruster, celle dans laquelle faire l'incrustation, les coordonnées des 4 points d'incrustation (nos picots ordonnés), ainsi qu'une matrice contenant les zones que l'on ne veut pas modifier (dans notre cas les doigts sur la vidéo).

FONCTION_ORDRE_PICOT

Cette fonction sert à ordonner les picots labellisés sur la première image. Pour cela on fixe un picot en tant que numéro 1, ensuite on effectue des comparaisons sur les produits vectoriels issus des autres picots en fonction du numéro 1 ; suivant les valeurs obtenues on détermine les numéros des autres picots.

FONCTION_TRACKINGS

Cette fonction prend en entrée la matrice des barycentres de l'image issue de la fonction *fonction_labellisation_picot* qui permet de segmenter les picots de l'image en cours ainsi que la matrice des barycentres ordonnés de l'image précédente. Pour chaque barycentre de l'image en cours, on cherche à faire correspondre ce dernier avec un des barycentres de l'image précédente. Pour cela, on calcule la distance (traduit par la norme dans R^2) entre le barycentre de l'image en cours et les barycentres de l'image précédente. On prend la distance minimale. On peut en déduire ainsi la position des picots au cours de la vidéo.

FONCTION_LABELLISATION

Cette fonction prend en entrée l'image seuillée montrant les picots. Du fait du choix du seuil, l'image n'est composé que de pixels nul ou de pixels dont la valeur est 1. De plus, pour les mêmes raisons, il n'y a que 4 zones composé de 1 correspondant aux positions des picots. On associe à chaque zone un nombre différent. On a donc 4 zones composées de pixels dont la valeur est 1, 2, 3 et 4 en utilisant la fonction *bwlabel.*

Pour chaque zone, on prend les positions [X Y] des pixels égaux à 1, 2, 3 ou 4 selon la zone. On calcule les moyennes de ces positions permettant ainsi de calculer les barycentres spatiaux des picots (qui sont virtuels).

RESULTAT OBTENU

On obtient après l'application de cet algorithme une vidéo avec une image incrustée, disponible dans l'archive. Le choix a été fait ici d'incruster différentes images au format .gif dans la vidéo. Le but était de pouvoir incruster plusieurs contenus dans la vidéo et non pas uniquement une image.

CONCLUSION

BILAN DES FONCTIONS MATLAB UTILISEES

mean, save, VideoReader, VideoWriter, bwlabel, find, sqrt, reshape,

PERSPECTIVES D'AMELIORATION

RECOUVREMENT TOTAL

L'un des premiers objectifs du programme serait ensuite de faire un recouvrement total de la feuille. En effet, l'algorithme se repérant par rapport aux quatre picots, il est impossible actuellement de couvrir complètement la feuille blanche mais uniquement le quadrangle délimité par les picots.

GENERALISATION A TOUTE LES FORMES

L'une des perspectives d'évolution du programme et d'être appliqué à des formes différentes du quadrangle qu'incarne la feuille. L'étape critique dans le cas d'une variation du nombre de coins (et donc du nombre de picots vient de la labellisation de ces derniers. Là on a préféré entrer manuellement la matrice d'ordonnancement des picots, on préférera développer une fonction récursive afin d'écrire de manière automatique la matrice d'ordonnancement.

OPTIMISATION DE LA SEGMENTATION DES DOIGTS

On remarque sur les doigts des discontinuités dans le sens où les doigts ne sont pas parfaitement segmentés

(Et ce depuis la définition du modèle des doigts). Cependant, l'application d'homographie pourrait être la solution à une meilleure définition des doigts. L'autre solution serait de faire une segmentation basé sur les contours (gradient, laplacien par exemple)

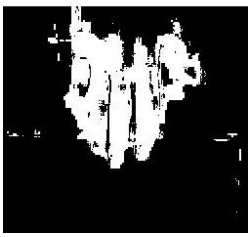


Fig. : Image seuillée (première apparition des doigts)

image finale de la vidéo

ANNEXES

CODE MATLAB

GENERATION_DU_MODELE.M

```
close all

%Sélection d'un picot
vid=VideoReader('vid_in.mp4');
img1=read(vid,1);
picot=double(img1(292:303,376:384,:));

%Calcul de la moyenne
Moyenne=fonction_moyennage(picot);

%Calcul covariance
MatCov=fonction_covariance(picot,Moyenne);

%Distances de Mahalanobis pour la première image
Mahalanobis=fonction_mahalanobis(img1,Moyenne,MatCov);

%Seuillage de la première image
Seuil=50;

%Même démarche que pour la détection des picots mais cette fois avec les
%doigts
imgdoigts=read(vid,38);
doigts=double(imgdoigts(108:127,276:292,:));
Moydoigts=fonction_moyennage(doigts);
MatCovDoigts=fonction_covariance(doigts,Moydoigts);
Mahalanobisdoigts=fonction_mahalanobis(imgdoigts,Moydoigts,MatCovDoigts);

% figure,imshow(Mahalanobisdoigts,hot);
Seuildoigts=30;
% imgsdoigts=fonction_seuillage(imgdoigts,Mahalanobisdoigts,Seuildoigts);
% figure,imshow(imgsdoigts);

%Sauvegarde du modèle
save('modele.mat','Seuil','MatCov','Mahalanobis','Moyenne','Seuildoigts','M
atCovDoigts','Mahalanobisdoigts','Moydoigts');
```

DETECTION_DES_PICOTS.M

```
clear all
close all

load('modele.mat');
video_origine=VideoReader('vid_in.mp4');
img=read(video_origine,1);
imgS=fonction_seuillage(img,Mahalanobis,Seuil);

MatBary=fonction_labellisation_picot(imgS);
MatBaryOrdonnee=fonction_ordre_picot(MatBary);

echelle=1;
mask = zeros(size(img,1),size(img,2));
[goku,map]=imread('goku.gif','frames','all');
goku1=ind2rgb8(goku(:,:,1),map);
```

```
num=1;
imgE=fonction_incrustation(goku1,img,MatBaryOrdonnee(2,:),MatBaryOrdonnee(1
,:),echelle,mask);
%figure,imshow(imgE);

video_finale=VideoWriter('video_finale.avi', 'Motion JPEG AVI');
video_finale.FrameRate=25;
open(video_finale);
writeVideo(video_finale,imgE);
for k=2:248
    img=read(video_origine,k);
    mahalanobis=fonction_mahalanobis(img,Moyenne,MatCov);
    imgS=fonction_seuillage(img,mahalanobis,Seuil);
    MatBary=fonction_labellisation_picot(imgS);
    MatBaryPrecedent=MatBaryOrdonnee;
    MatBaryOrdonnee=fonction_tracking(MatBary,MatBaryPrecedent);

    if (k>=38)
    imgdoigts=read(video_origine,k);

Mahalanobisdoigts=fonction_mahalanobis(imgdoigts,Moydoigts,MatCovDoigts);
    imgsdoigts=fonction_seuillage(imgdoigts,Mahalanobisdoigts,Seuildoigts);
    mask=imgsdoigts;
    end

    if (k<50)
    if num<=5
        num=num+1;
    else
        num=0;
    end
    [goku,map]=imread('goku.gif','frames','all');
    goku1=fonction_gif0(num,goku,map);

imgE=fonction_incrustation(goku1,img,MatBaryOrdonnee(2,:),MatBaryOrdonnee(1
,:),echelle,mask);



    elseif (k<100)
    if num<=10
        num=num+1;
    else
        num=0;
    end
    [goku,map]=imread('gokussj1.gif','frames','all');
    goku1=fonction_gif1(num,goku,map);

imgE=fonction_incrustation(goku1,img,MatBaryOrdonnee(2,:),MatBaryOrdonnee(1
,:),echelle,mask);



    elseif (k<145)
    if num<=7
        num=num+1;
    else
        num=0;
    end
    [goku,map]=imread('gokussj3.gif','frames','all');
```

```
    goku1=fonction_gif2(num,goku,map);

imgE=fonction_incrustation(goku1,img,MatBaryOrdonnee(2,:),MatBaryOrdonnee(1
,:),echelle,mask);



    elseif (k<=200)
    if num<=8
        num=num+1;
    else
        num=0;
    end
    [goku,map]=imread('gokugod1.gif','frames','all');
    goku1=fonction_gif3(num,goku,map);

imgE=fonction_incrustation(goku1,img,MatBaryOrdonnee(2,:),MatBaryOrdonnee(1
,:),echelle,mask);



    else
    if num<=10
        num=num+1;
    else
        num=0;
    end
    [goku,map]=imread('gokugodssj.gif','frames','all');
    goku1=fonction_gif1(num,goku,map);

imgE=fonction_incrustation(goku1,img,MatBaryOrdonnee(2,:),MatBaryOrdonnee(1
,:),echelle,mask);
    end

    writeVideo(video_finale,imgE);
end
close(video_finale);
```

FONCTION_GIF0.M

```
function [ goku1] = fonction_gif0( num , goku , map)
if num==0
goku1=ind2rgb8(goku(:,:,1),map);
elseif num==1
goku1=ind2rgb8(goku(:,:,2),map);
elseif num==2
goku1=ind2rgb8(goku(:,:,3),map);
elseif num==3
goku1=ind2rgb8(goku(:,:,4),map);
elseif num==4
goku1=ind2rgb8(goku(:,:,5),map);
elseif num==5
goku1=ind2rgb8(goku(:,:,6),map);
elseif num==6
goku1=ind2rgb8(goku(:,:,7),map);
end
end
```

FONCTION_GIF1.M

```
function [ goku1] = fonction_gif1( num , goku , map)
if num==0
goku1=ind2rgb8(goku(:,:,1),map);
elseif num==1
goku1=ind2rgb8(goku(:,:,2),map);
elseif num==2
goku1=ind2rgb8(goku(:,:,3),map);
elseif num==3
goku1=ind2rgb8(goku(:,:,4),map);
elseif num==4
goku1=ind2rgb8(goku(:,:,5),map);
elseif num==5
goku1=ind2rgb8(goku(:,:,6),map);
elseif num==6
goku1=ind2rgb8(goku(:,:,7),map);
elseif num==7
goku1=ind2rgb8(goku(:,:,8),map);
elseif num==8
goku1=ind2rgb8(goku(:,:,9),map);
elseif num==9
goku1=ind2rgb8(goku(:,:,10),map);
elseif num==10
goku1=ind2rgb8(goku(:,:,11),map);
elseif num==11
goku1=ind2rgb8(goku(:,:,12),map);
end
end
```

FONCTION_GIF2.M

```
function [ goku1] = fonction_gif2( num , goku , map)
if num==0
goku1=ind2rgb8(goku(:,:,1),map);
elseif num==1
goku1=ind2rgb8(goku(:,:,2),map);
elseif num==2
goku1=ind2rgb8(goku(:,:,3),map);
elseif num==3
goku1=ind2rgb8(goku(:,:,4),map);
elseif num==4
goku1=ind2rgb8(goku(:,:,5),map);
elseif num==5
goku1=ind2rgb8(goku(:,:,6),map);
elseif num==6
goku1=ind2rgb8(goku(:,:,7),map);
elseif num==7
goku1=ind2rgb8(goku(:,:,8),map);
elseif num==8
goku1=ind2rgb8(goku(:,:,9),map);
end
end
```

FONCTION_GIF3.M

```
function [ goku1] = fonction_gif3( num , goku , map)
if num==0
goku1=ind2rgb8(goku(:,:,1),map);
elseif num==1
```

```
goku1=ind2rgb8(goku(:,:,2),map);
elseif num==2
goku1=ind2rgb8(goku(:,:,3),map);
elseif num==3
goku1=ind2rgb8(goku(:,:,4),map);
elseif num==4
goku1=ind2rgb8(goku(:,:,5),map);
elseif num==5
goku1=ind2rgb8(goku(:,:,6),map);
elseif num==6
goku1=ind2rgb8(goku(:,:,7),map);
elseif num==7
goku1=ind2rgb8(goku(:,:,8),map);
elseif num==8
goku1=ind2rgb8(goku(:,:,9),map);
elseif num==9
goku1=ind2rgb8(goku(:,:,10),map);
elseif num==10
goku1=ind2rgb8(goku(:,:,11),map);
elseif num==11
goku1=ind2rgb8(goku(:,:,12),map);
elseif num==12
goku1=ind2rgb8(goku(:,:,13),map);
elseif num==13
goku1=ind2rgb8(goku(:,:,14),map);
end
end
```

FONCTION_COVARIANCE.M

```
function [ MatCov ] = fonction_covariance(img,moy)
R=img(:,:,1); %on détermine la composante R de notre région
G=img(:,:,2); %on détermine la composante G de notre région
B=img(:,:,3); %on détermine la composante B de notre région
%DEFINITION DE LA MATRICE DE COVARIANCE
C11=(R-moy(1)).*(R-moy(1));
CRR=mean(C11(:));
C12=(R-moy(1)).*(G-moy(2));
CRG=mean(C12(:));
C13=(R-moy(1)).*(B-moy(3));
CRB=mean(C13(:));
CGR=CRG;
C22=(G-moy(2)).*(G-moy(2));
CGG=mean(C22(:));
C23=(G-moy(2)).*(B-moy(3));
CGB=mean(C23(:));
CBR=CRB;
CBG=CGB;
C33=(B-moy(3)).*(B-moy(3));
CBB=mean(C33(:));

MatCov=[CRR CRG CRB; CGR CGG CGB; CBR CBG CBB];

end
```

FONCTION_MOYENNE.M

```
function [ moy ] = fonction_moyennage( img )
```

```
R=img(:,:,1); %on détermine la composante R de notre région
G=img(:,:,2); %on détermine la composante G de notre région
B=img(:,:,3); %on détermine la composante B de notre région

moyR=mean(R(:)); %Moyenne de la composante R de notre région
moyG=mean(G(:)); %Moyenne de la composante G de notre région
moyB=mean(B(:)); %Moyenne de la composante B de notre région
moy=[moyR,moyG,moyB]; %On stoque les moyennes dans un vecteur moyenne
commun

end
```

FONCTION_ORDRE_PICOT.M

```
function [ MatBaryOrdonnes ] = fonction_ordre_picot(MatBary)



%Création des 3 vecteurs à partir du point 1 fixé
bary1final=[MatBary(1,4);MatBary(2,4)];
V12=[MatBary(1,2)-MatBary(1,4);MatBary(2,2)-MatBary(2,4)];
V13=[MatBary(1,1)-MatBary(1,4);MatBary(2,1)-MatBary(2,4)];
V14=[MatBary(1,3)-MatBary(1,4);MatBary(2,3)-MatBary(2,4)];

%Calculs des produits vectoriels
VectProduit=[V12(1)*V13(2)-V12(2)*V13(1) V12(1)*V14(2)-V12(2)*V14(1)
V13(1)*V14(2)-V13(2)*V14(1)];

if (VectProduit(1)>0)
    if (VectProduit(2)>0)
        if (VectProduit(3)>0)
            bary2final=[MatBary(1,2);MatBary(2,2)];
            bary3final=[MatBary(1,1);MatBary(2,1)];
            bary4final=[MatBary(1,3);MatBary(2,3)];
        else
            bary2final=[MatBary(1,2);MatBary(2,2)];
            bary3final=[MatBary(1,3);MatBary(2,3)];
            bary4final=[MatBary(1,1);MatBary(2,1)];
        end
    else
            bary2final=[MatBary(1,3);MatBary(2,3)];
            bary3final=[MatBary(1,2);MatBary(2,2)];
            bary4final=[MatBary(1,1);MatBary(2,1)];
    end
else
    if (VectProduit(2)>0)
            bary2final=[MatBary(1,1);MatBary(2,1)];
            bary3final=[MatBary(1,2);MatBary(2,2)];
            bary4final=[MatBary(1,3);MatBary(2,3)];
    else
        if (VectProduit(3)>0)
            bary2final=[MatBary(1,1);MatBary(2,1)];
            bary3final=[MatBary(1,3);MatBary(2,3)];
            bary4final=[MatBary(1,2);MatBary(2,2)];
        else
            bary2final=[MatBary(1,3);MatBary(2,3)];
            bary3final=[MatBary(1,1);MatBary(2,1)];
```

```
            bary4final=[MatBary(1,2);MatBary(2,2)];
        end
    end
end

MatBaryOrdonnes=[bary1final bary2final bary3final bary4final];
end
```

FONCTION_SEUILLAGE.M

```
function [ImgS]=fonction_seuillage(imgI,mahalanobis,Seuil)
ImgS=ones(size(imgI,1),size(imgI,2));
for k=1:size(imgI,1)
    for i=1:size(imgI,2)
        if (mahalanobis(k,i)>Seuil)
            ImgS(k,i)=0;
        end
    end
end
```

FONCTION_MAHALANOBIS.M

```
function [ Mahalanobis] = fonction_mahalanobis(img,moy,MatCov)
h=size(img,1);
w=size(img,2);
nbpixels=h*w;
MatCovInv=(MatCov^(-1));
imgdouble=double(img);
imgReshaped=reshape(imgdouble,[],3,1)-repmat(moy,nbpixels,1);
imgMaha=sum((MatCovInv*imgReshaped').*imgReshaped',1);
Mahalanobis=reshape(imgMaha,h,w);
end
```

FONCTION_LABELLISATION.M

```
function [ MatBary ] = fonction_labellisation_picot( ImgSeuillee )
%imerode
ImgSL=logical(ImgSeuillee);
[L,num]=bwlabel(ImgSL);

%labellisation des 4 picots et calcul des barycentres

MatBary=[2 double(num)];
for i=1:num
    [r, c] = find(L==i);
    MatBary(1,i) = (mean(r));
    MatBary(2,i)=(mean(c));
end
end
```

FONCTION_TRACKING.M

```
function [ MatBaryOrdonnes ] = fonction_tracking( MatBary,MatBaryOrdonnes )
%Tracking des picots sur les autres images
MatriceTracking=zeros(4,size(MatBary,2));
MatriceTracking(1,:)=sqrt((MatBary(1,:)-
MatBaryOrdonnes(1,1)).^2+(MatBary(2,:)-MatBaryOrdonnes(2,1)).^2);
```

```
MatriceTracking(2,:)=sqrt((MatBary(1,:)-
MatBaryOrdonnes(1,2)).^2+(MatBary(2,:)-MatBaryOrdonnes(2,2)).^2);
MatriceTracking(3,:)=sqrt((MatBary(1,:)-
MatBaryOrdonnes(1,3)).^2+(MatBary(2,:)-MatBaryOrdonnes(2,3)).^2);
MatriceTracking(4,:)=sqrt((MatBary(1,:)-
MatBaryOrdonnes(1,4)).^2+(MatBary(2,:)-MatBaryOrdonnes(2,4)).^2);
[Pmin,loc]=min(MatriceTracking,[],2);
MatBaryOrdonnes=[MatBary(:,loc(1)) MatBary(:,loc(2)) MatBary(:,loc(3))
MatBary(:,loc(4))];
end
```

FONCTION_INCRUSTATION.M

```
function frame=fonction_incrustation(motif,frame,x,y,scale,mask)
    % motif : image 'source' (couleur indexée ou vraie couleur)
    % frame : image 'destination' (vraie couleur)
    % x,y   : coordonnées des 4 sommets de la 'source' dans la
'destination', vecteurs lignes
    % scale : paramètre d'échelle (exemple : 1)
    % mask  : masque 'destination' des pixels à ne pas modifier, (exemple :
matrice de 0)

    [hIn,wIn,dIn]=size(motif);
    xIn=[1 wIn wIn 1];
    yIn=[1 1 hIn hIn];
    xIn=wIn/2+scale*(xIn-wIn/2);
    yIn=hIn/2+scale*(yIn-hIn/2);
    tForm=cp2tform([xIn' yIn'],[x' y'],'projective');
    motif=double(motif);
    for p=1:3
        if dIn == 1

[motifTransform,xData,yData]=imtransform(motif(:,:),tForm,'Fill',-1);
        else

[motifTransform,xData,yData]=imtransform(motif(:,:,p),tForm,'Fill',-1);
        end
        [hOut,wOut]=size(motifTransform);
        xOut=fix(xData(1));
        yOut=fix(yData(1));
        dxOut=xOut:xOut+wOut-1;
        dyOut=yOut:yOut+hOut-1;
        pos=find(mask(dyOut,dxOut)==1);
        if (length(pos))
            motifTransform(pos)=-1;
        end
        pos=find(motifTransform~=-1);
        frameCut=frame(dyOut,dxOut,p);
        if (length(pos))
            frameCut(pos)=uint8(motifTransform(pos));
        end
        frame(dyOut,dxOut,p)=frameCut;
    end
end
```

SCHEMA BLOC : DETERMINATION DU MODELE DES REPERES BLEUS