2016

CO7SFTS0 – Projet de réalité augmentée

Lemaire de Mil Arthur

Lille Laura

Mounaix Marion

Bordeaux INP - ENSC

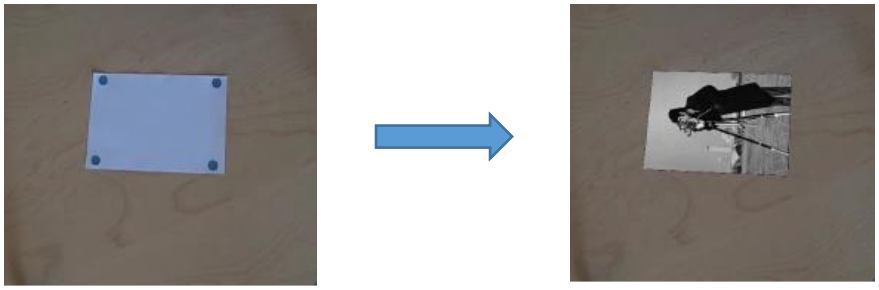
12/12/2016

Table des matières

Introduction

De nombreux logiciels proposent la possibilité de faire du remplacement de contenu comme par exemple Photoshop ou VirtualDub.
Le but de ce projet était de réaliser une insertion d'une image dans une vidéo avec MatLab.
La vidéo présentait une main déplaçant un rectangle bleu clair (contenant 4 picots bleus en ses sommets) sur une table, l'image devait donc prendre la place du rectangle.
La stratégie aurait été de détecter l'affiche bleu clair pour projeter l'image dessus, or il y a des amas de bleu clair, rendant difficile une détection « propre » du rectangle.
Il a donc fallu se servir des picots présents. Plusieurs problématiques ont alors émergé :

- Comment suivre la trajectoire du rectangle?
- Comment effectuer de la segmentation couleur (utilisation des picots)?
- Comment incruster et adapter l'image au rectangle?



ETAT INITIAL ETAT FINAL

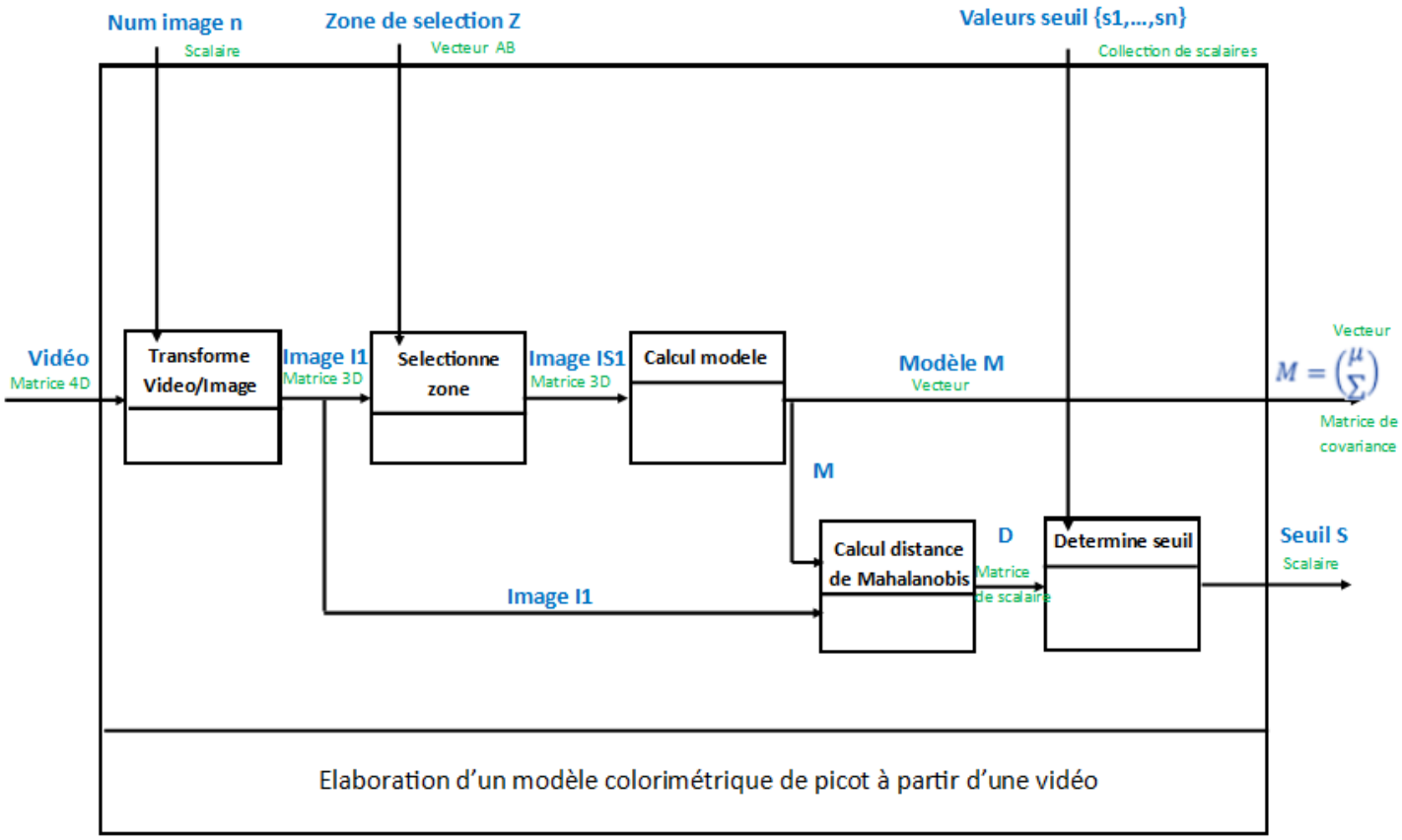
I - Démarche algorithmique

La démarche algorithmique s'est décomposée en trois parties :

a) Identification du modèle

La première étape du projet est l'élaboration d'un modèle colorimétrique de picot.

Cette étape est représentée par le schéma bloc ci-dessous :



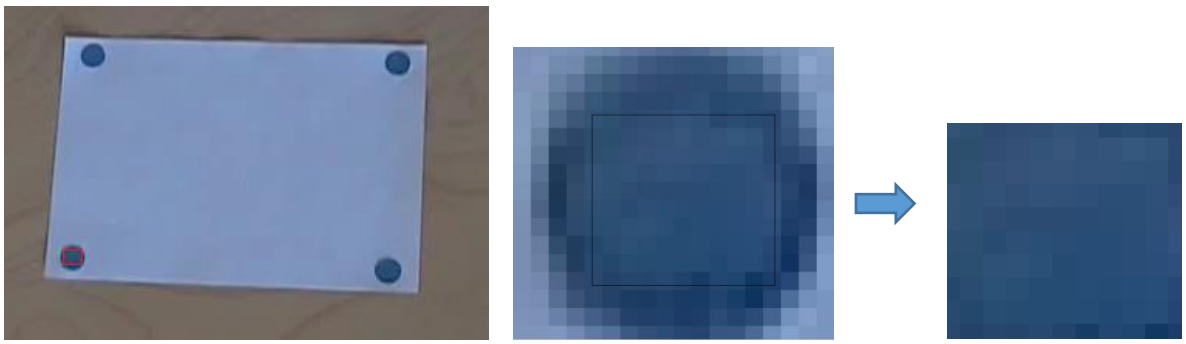
Bloc « Transforme Video/Image »

Il fallait tout d'abord lire la vidéo et la décomposer en images. Nous nous sommes intéressés à la première image pour établir notre modèle.

Ensuite venait la segmentation de couleurs, ie l'extraction d'attributs caractérisant les entités picots bleus.

Bloc « Selectionne zone »

Il fallait sélectionner une région d'intérêt rectangulaire dans un des picots, comme présenté ci-dessous :



Bloc « Calcul modèle »

Avec cette zone, nous avons pu calculer le modèle, composé d'un vecteur µ comportant les valeurs moyennes des composantes bleu, vert et rouge de chaque pixel d'un picot ; et une matrice de covariance Σ. Voici leur formule théorique :

$$\mu = \begin{bmatrix} \mu^1 \\ \mu^2 \end{bmatrix} = \overline{x_i} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \Sigma = \begin{pmatrix} \Sigma^{1,1} & \Sigma^{1,2} \\ \Sigma^{2,1} & \Sigma^{2,2} \end{pmatrix} = \frac{1}{N}\begin{pmatrix} \sum_{i=1}^{N}(x_i^1-\mu^1)(x_i^1-\mu^1) & \sum_{i=1}^{N}(x_i^1-\mu^1)(x_i^2-\mu^2) \\ \sum_{i=1}^{N}(x_i^1-\mu^1)(x_i^2-\mu^2) & \sum_{i=1}^{N}(x_i^2-\mu^2)(x_i^2-\mu^2) \end{pmatrix}$$

Nous avons obtenu le modèle suivant :

```
>> Initialisation

mu =

   42.8081   77.1212  116.3232


S =

   1.0e+03 *

    4.4074    3.1693    2.2931
    3.1693    2.8065    2.1141
    2.2931    2.1141    2.4837
```

Il fallait enfin déterminer un seuil pour détecter les picots bleus.

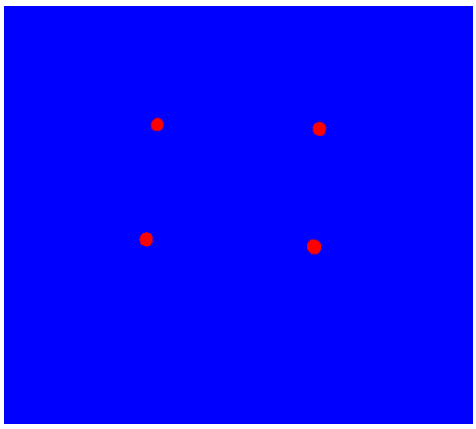
Bloc « Calcul distance de Mahalanobis »

Pour cela, nous avons calculé la distance de segmentation (distance entre un pixel et un modèle établi auparavant), voici sa formule :

$$D^{Maha}(x_i) = (x_i - \mu)^T \Sigma^{-1} (x_i - \mu)$$

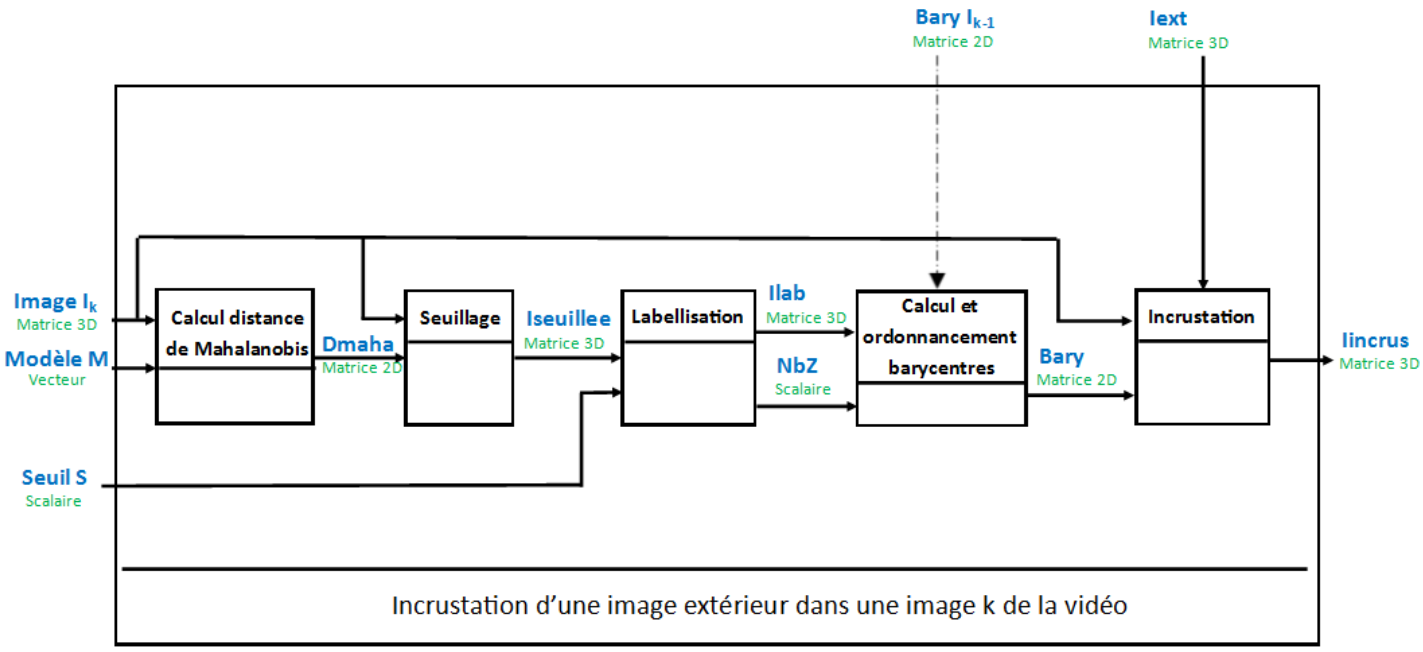
Bloc « Determine seuil »

Puis, en effectuant plusieurs tests, on a établi Vseuil = 0.07, permettant d'isoler proprement les quatre picots.



b) Remplacement partiel de la vidéo

Cette étape peut être représentée par le schéma bloc ci-dessous (à noter que pour la première image, la flèche en pointillé n'est pas présente) :

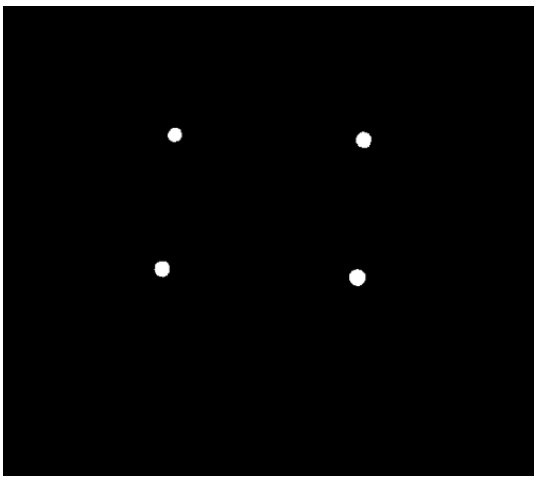


Nous avons **effectué ce schéma bloc n fois**, avec n le nombre d'images dans la vidéo.

Pour le remplacement, il fallait tout d'abord effectuer le seuillage de chaque image.

Bloc « Calcul distance de Mahalanobis »
Pour cela, comme précédemment, calculer la distance de Mahalanobis.

Bloc « Seuillage »
On obtient alors Iseuillee, c'est l'image I_k avec les picots qui sont devenus blancs.



Bloc « Labellisation »

La labellisation sert à séparer une image en zones.

Avec Matlab, on utilise la fonction *bwlabel* :

Create a small binary image.

```
BW = logical ([1     1     1     0     0     0     0     0
               1     1     1     0     1     1     0     0
               1     1     1     0     1     1     0     0
               1     1     1     0     0     0     1     0
               1     1     1     0     0     0     1     0
               1     1     1     0     0     0     1     0
               1     1     1     0     0     1     1     0
               1     1     1     0     0     0     0     0]);
```

Create the label matrix using 4-connected objects.

```
L = bwlabel(BW,4)
```

```
L =

     1     1     1     0     0     0     0     0
     1     1     1     0     2     2     0     0
     1     1     1     0     2     2     0     0
     1     1     1     0     0     0     3     0
     1     1     1     0     0     0     3     0
     1     1     1     0     0     0     3     0
     1     1     1     0     0     3     3     0
     1     1     1     0     0     0     0     0
```

Le principe : on compare la distance de Mahalanobis de chaque pixel au seuil ; si Dmaha>Seuil, alors ce pixel n'appartient pas au picot bleu.

On obtient Ilab, une image binaire avec des étiquettes (numéros des objets trouvés).

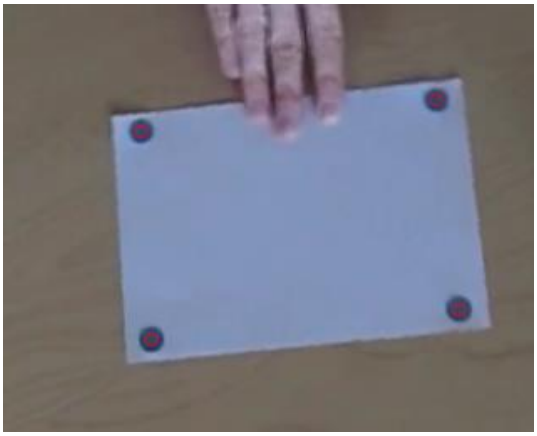
Il faut maintenant calculer les quatre centres virtuels des picots.

Bloc « Calcul et ordonnancement barycentres »

Pour la première image de la vidéo, on va définir un ordre de barycentre.

Pour les images suivantes, on va devoir éliminer les barycentres en trop pour avoir les quatre. On va pour cela comparer aux barycentres de l'image précédente, en trouvant les distances minimales.

On obtient ainsi les quatre barycentres des picots :

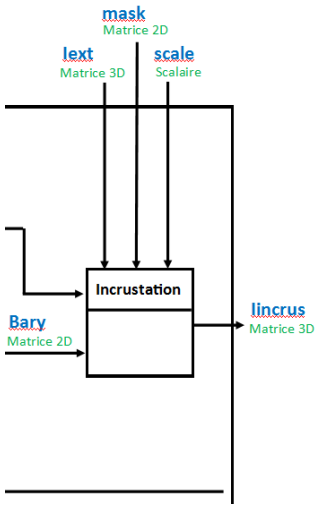


Bloc « Incrustation »
On a un quadrangle avec les quatre barycentres, on peut ainsi incorporer l'image Iext à l'image I_k de la vidéo.

Ainsi, l'image sera un peu plus petite que le rectangle bleu clair :

c) Remplacement total

Pour un remplacement total, il suffit d'ajouter un facteur d'agrandissement (scale) de 0,80.
De plus, pour ne pas cacher la main, on ajouter un masque (mask). Le masque est créé de la même manière que le modèle des picots mais avec les pixels des doigts.
Le schéma bloc change légèrement sur la fin :



En réitérant, on obtient la vidéo souhaitée.

II – Implémentation

Afin de résoudre le problème nous avons commencé par utiliser une fonction permettant d'isoler le modèle :

Initialisation.m

Tout d'abord on importe la vidéo dans la variable vidéo.

Pour isoler le modèle, une seule image est nécessaire, on récupère donc la première image de la vidéo dans la variable I. On utilise ensuite l'outil de Matlab *imshow* afin de visionner I , et la fonction *ginput* afin de récupérer les coordonnées du rectangle, défini par les deux points sélectionnés. On va ensuite superposer ce rectangle à I afin de récupérer une petite image ne contenant que la zone sélectionnée : IS.

Afin de trouver mu, on isole les composantes rouges, vertes et bleues de IS dans 3 nouvelles images (respectivement ISR, ISV, ISB). On calcule ensuite la valeur moyenne (à l'aide de la fonction *mean*) de chaque composante que l'on stocke dans le tableau mu.

On cherche ensuite à déterminer les valeurs de la matrice S, on calcule donc les sommes pour chaque terme de la matrice (ou presque, S(2,1)=S(1,2), S(3,1)=S(1,3) et S(3,2)=S(2,3)). On récupère ces valeurs dans la matrice 3x3 S. On obtient ainsi le modèle M = [mu;S].

On cherche ensuite à calculer la distance de mahalanobis en chaque point de I, que l'on stocke dans DmalaI, on fait pour cela appel à la fonction *mahalanobis.*

mahalanobis.m

Arguments:

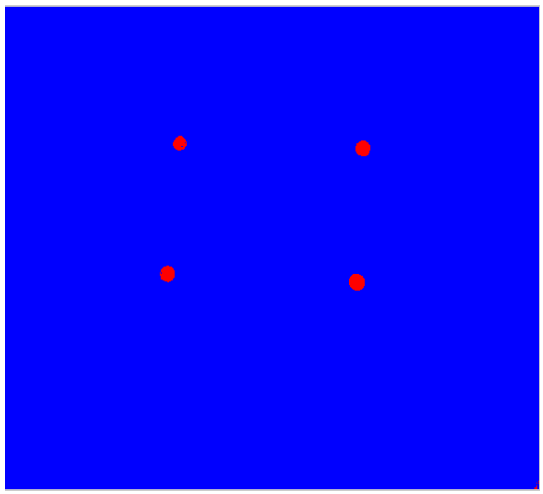
- img1 : qui correspond à l'image sur laquelle on veut calculer la distance de mahalanobis
- moy : qui correspond au mu
- mat_cov : qui correspond à S

On commence par étudier l'image I en récupérant sa longueur (h), sa largeur(l) et son nombre de pixels (nbp). On crée ensuite la nouvelle image img2 qui recevra les valeurs de la distance de mahalanobis de chaque point de l'image.

On place ensuite « bout à bout » les trois composantes de img2 grâce à la fonction *reshape* qui nous permet ici de transformer une matrice à 3 dimensions en un vecteur colonne que l'on stocke dans img2. On soustrait à img2 un vecteur de même dimension ne contenant que la valeur de mu. On applique ensuite la formule de mahalanobis à img2 et mat_cov. Ce qui nous permet d'obtenir la matrice contenant les distances de mahalanobis entre chaque point de I et le modèle.

Initialisation.m

On initialise Vseuil et I2 (que l'on crée aux mêmes dimensions que I et que l'on remplit de 1). On parcourt ensuite DmalaI pixel par pixel (grâce à une double boucle) et on compare la valeur du pixel à Vseuil : si le pixel est inférieur à Vseuil, on le colore en rouge dans I2, sinon on le colore en bleu. On obtient une image du type :



On a donc un bon modèle et un bon Vseuil, que l'on enregistre dans un fichier **modele.mat.**

Suivi.m

On commence par importer la vidéo dans video, le modèle et l'image que l'on souhaite insérer dans la vidéo dans Imotif.

On entre ensuite dans une boucle qui va parcourir video image par image. On calcule donc la distance de mahalanobis de la $i^{ème}$ image de video que l'on place dans DmalaI.

On cherche ensuite à seuiller DmalaI, pour cela on trouve le max de DmalaI (IMax) et on divise chaque point de DmalaI par IMax afin d'avoir pour chaque pixel une valeur entre 0 et 1. On compare ensuite chaque pixel à Vseuil et on obtient l'image seuillée ISeuillee.
On applique ensuite des opérateurs morphologiques à ISeuillee (grâce aux fonctions imerode et imdilate), on utilise un voisinage arbitraire (fonction strel) de 50pixels.

On utilise ensuite la fonction bwlabel afin de labelliser ISeuillee, on trouve ensuite les barycentres de l'image labélisée grâce à la fonction Bary.m.

Bary.m

La fonction *Bary* prend en arguments une image labélisée (L) et le nombre de zones labélisées (nbrzone).

On commence par extraire les coordonnées de chaque point de la zone labélisée a l'aide de la fonction *find*. On calcule ensuite la moyenne des coordonnées en abscisse et en ordonnée et on la stocke dans un tableau. Cette action est réalisée pour chaque zone labélisée de l'image.

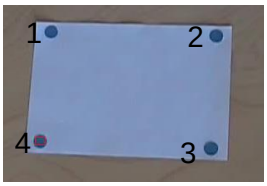
On obtient en sortie un tableau ayant le format suivant :

nbzone

x_1	x_2	...	x_{nbzone}
y_1	y_2	...	y_{nbzone}

Suivi.m

On va ensuite ordonner les barycentres. Pour cela on commence par vérifier si on traite la première image de la vidéo. Si c'est le cas, on a décidé de placer en premier le barycentre en haut à gauche et de tourner dans le sens horaire (cette décision est arbitraire). On les ordonne ensuite grâce à leurs coordonnées et on récupère le tableau suivant :



x_4	x_1	x_2	x_3
y_4	y_1	y_2	y_3

En revanche, si on ne traite pas la première image, on ordonne les barycentres en fonction de l'image précédent à l'aide des fonctions *distance* et *TrouveBary*.

Distance.m

La fonction *distance* prend en argument deux tableaux de barycentres (*oldB* et *newB*, qui contiennent respectivement les barycentres de l'image précédente et les barycentres de l'image que l'on traite) et permet d'obtenir le tableau suivant :

	old1	old2	old3	old4
new1	Dist(old1,new1)	Dist(old2,new1)	Dist(old3,new1)	Dist(old4,new1)
new2	Dist(old1,new2)	Dist(old2,new2)	Dist(old3,new2)	Dist(old4,new2)
new3	Dist(old1,new3)	Dist(old2,new3)	Dist(old3,new3)	Dist(old4,new3)
...	...	...	...	...
newi	Dist(old1,newi)	Dist(old2,newi)	Dist(old3,newi)	Dist(old4,newi)

Où si a=(x_a,y_a) et b=(x_b,y_b),

$$Dist(a,b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

On obtient donc la distance entre chaque barycentre de l'image traitée et de l'image précédente. Ces derniers sont ensuite transmis à la fonction *TrouveBary*.

TrouveBary.m

La fonction *TrouveBary* prend en argument une matrice (distance) de distance et un tableau de barycentres (newB) et permet d'obtenir un tableau de 4 barycentres ordonnés.

La fonction TrouveBary parcourt toute la matrice distance et trouve la plus petite valeur et enregistre ses coordonnées dans les variables Xmin et Ymin. Elle remplace ensuite toutes les valeurs de la ligne et de la colonne par -1.

Les valeurs des exemples ci-dessous sont des valeurs fictives :

	old1	old2	old3	old4
new1	280.56	340.30	250.21	0.44
new2	0.39	210.34	180.29	98.3
new3	280.31	340.12	250.11	0.42
new4	95.32	100.00	0.21	230.12
new5	190.30	0.12	240.31	94.37

Xmin=4

Ymin=2

	old1	old2	old3	old4
new1	280.56	-1	250.21	0.44
new2	0.39	-1	180.29	98.3
new3	280.31	-1	250.11	0.42
new4	95.32	-1	0.21	230.12
new5	-1	-1	-1	-1

On va ensuite prendre les valeurs dans la Xmin^{ième} colonne du tableau newB et la placer dans la Ymin^{ième} colonne de notre nouveau tableau de barycentre (coordonnee).

NewB

123.45	124.09	350.29	361.43
134.56	329.45	332.89	135.30

Coordonnee

	361.43		
	135.30		

On répète ensuite cette opération quatre fois (soit jusqu'à ce que la matrice distance soit remplie de -1), on s'assure de cette manière d'avoir 4 barycentres en sortie et qu'ils correspondent bien aux nouvelles positions centrales des points bleus sur l'image.

Suivi.m

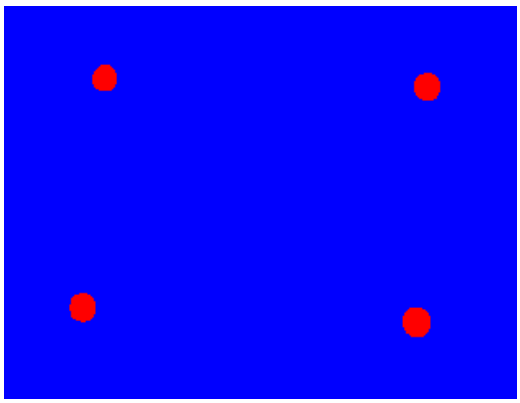
On récupère ensuite les nouveaux barycentres dans newBaryOrdonnes, on utilise ensuite la fonction *motif2frame* afin d'incruster l'image Imotif choisie dans l'image Ivideo.
On ajoute ensuite l'image frame à la vidéo grâce à la fonction *wirteVideo*.

On répète cette opération pour les 248 images de la vidéo d'origine.

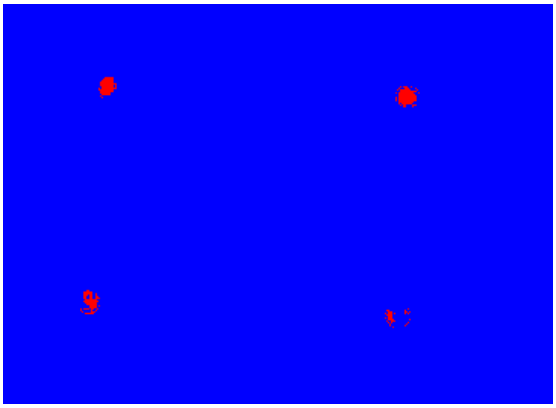
III - Résultats/Performance

Résultats :

Durant la partie d'initialisation, lorsque nous devons sélectionner le rectangle de pixels bleu, il faut le sélectionner dans le picot en bas à gauche de l'image. En effet, les couleurs des quatre picots varient à cause de l'éclairage de la vidéo. Avec le seuil que nous avons choisi, c'est le picot nous offrant le meilleur résultat :



Si la sélection est effectuée avec les autres picots, les résultats sont de ce type :

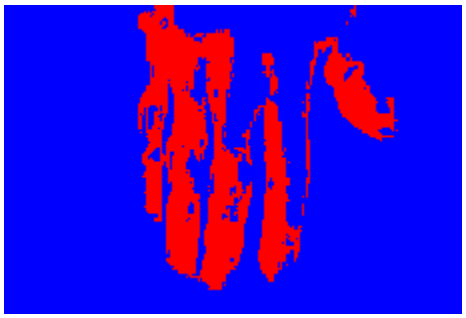


Juste avant la labellisation, nous avons utilisé des opérateurs morphologiques grâce aux fonctions *imerode* (érosion) et *imdilate* (dilatation). Ce sont des techniques de base appartenant au domaine de la morphologie et visant à filtrer une image. L'utilisation des opérateurs morphologiques permettent de filtrer les pixels incorrects dans l'image seuillée.

Une fois l'intégration de l'image réussie, il nous a fallu rajouter les doigts qui se retrouvaient cachés par l'image ajoutée.
Pour pallier à cela, nous avons réalisé un modèle des doigts. Il nous a donc fallu réutiliser notre fonction initialisation afin de réaliser un modèle des doigts.

Nous avons ensuite testé ce modèle sur différentes valeurs de seuil afin d'obtenir une bonne sélection de la main.

Nous avons obtenu ceci :



Nous pouvons remarquer que la détection de la main n'est pas parfaite, mais nous n'avons pas réussi à avoir un meilleur rendu.
Nous avons ensuite pu le superposer à l'image ajoutée en l'insérant comme mask dans l'utilisation de la fonction *motif2frame.*

Performances :

Lors du développement du programme, nous avons commencé par coder une première version de la fonction calculant la distance de mahalanobis, différente de celle décrite dans la partie II. La première fonction calculant la distance de mahalanobis que nous avons réalisé s'appelait dmala.

Dmala.m

La fonction prend en argument une image I, un vecteur mu et une matrice de covariance sigma.
La fonction parcourt l'image point par point à l'aide d'une double boucle et elle calcule pour chaque point sa distance de mahalanobis (Cf I pour la formule).

La fonction mahalanobis se réalise en O.n alors que la première version (dmalaI) se réalisait en $O.n^2$.
Ce qui explique la différence de performance entre les deux fonctions. En effet, le temps d'exécution de la nouvelle fonction est de 1 minute et 2 secondes, alors que l'ancienne prenait 40 minutes.

Conclusion

Ainsi, nous avons réussi à incorporer une image dans une vidéo. Reste encore quelques améliorations pour que l'intégration soit parfaite.

Avec ce projet, nous avons mis en pratique de la gestion de projet : nous avons dû segmenter le projet en étapes, penser d'abord le processus pour coder ensuite. Egalement, nous avons séparé les tâches entre nous trois, et nous avons dû nous concerter sur l'avancée du projet.
Nous avons aussi pris en main Matlab et commencer vraiment à manipuler cet outil à travers de ce projet.
Nous avons vu en pratique une partie du champ des possibles qu'offre le logiciel et pu observer un résultat concret : la production de la vidéo en réalité augmentée.

Annexe

Initialisation.m

```
%CREATION DU MODELE

clear all
close all

% Transfert video à image
video = VideoReader ('vid_in.mp4'); %
lire la vidéo
n = 1;
I = double(read (video, n)); %
récupére la n image de la vidéo
imshow(uint8(I)) % affiche l'image

% Selection zone
zoom on; % active le zoom
pause(); % enleve le zoom des qu'une
touche est appuyée
nb_points = 2;
[x,y] = ginput (nb_points); % permet
de trouver les abscisses et ordonnées
des 2 points cliqués

selection = [x(1),y(1),x(2)-x(1),y(2)-
y(1)]; % correspond à x0 y0 (point bas
gauche) longueur largeur
rectangle('Position', selection
)%tracer le rectangle
IS = double(imcrop (I,selection)); %
extraction de la sélection
figure(2), imshow(uint8(IS)) % affiche
la zone selectionnée

% Calcul modele
    % calcul de la moyenne mu
    ISR = IS(:,:,1); % matrice de la
composante rouge de IS
    R = ISR(:); % ISR sous forme de
vecteur colonne
    muR = mean(R); % moyenne de la
composante rouge de IS
    ISV = IS(:,:,2); % matrice de la
composante verte de IS
    V = ISV(:);
    muV = mean(V);
    ISB = IS(:,:,3); % matrice de la
composante bleue de IS
    B = ISB(:);
    muB = mean(B);

    mu = [muR muV muB];
    % muBis = mean(mean (IS))%
Remarque : cettte méthode ne
fonctionne que pour un rectangle

    % calcul de la matrice de
covariance Sigma
    S11 = sum((R-muR).*(R-muR)); % R -
muR correspond à un vecteur colonne
contenant (xiR - muR)²
    S12 = sum((R-muR).*(V-muV));
    S13 = sum((R-muR).*(B-muB));
    S21 = S12; % car les termes
antidiagonaux sont égaux
    S22 = sum((V-muV).*(V-muV));
    S23 = sum((V-muV).*(B-muB));
    S31 = S13;
    S32 = S23;
    S33 = sum((B-muB).*(B-muB));

    S = [S11 S12 S13; S21 S22 S23;
S31 S32 S33];


M = [mu;S]; % le modele M


%Calcul de la distance de Mahalanobis
DmalaI=mahalanobis(I,mu,S); %On
appelle la fonction exterieure dmala

%Determination du seuil
Vseuil=0.07; %rond en bas gauche

[l,c,a]=size(I);
I2=ones(l,c,a); %Création d'une image
vide
for i=1:l
        for j=1:c
            if (DmalaI(i,j)<Vseuil)
                I2(i,j,1)=255;
                I2(i,j,2)=0;
                I2(i,j,3)=0;
            else
                I2(i,j,1)=0;
                I2(i,j,2)=0;
                I2(i,j,3)=255;
            end
        end
end
imshow(uint8(I2));

% Sauvegarde du modèle et du seuil
save( 'modele.mat', 'M
```

mahalanobis.m

```
function [maha]=mahalanobis(image, moyenne, covariences)
    [h,l,a]=size(image);
    nbp=h*l;% calcule le nombre de pixels
    covariences_inv=(covariences^(-1));% calcule la transposée de la matrice de
covariance
    image2=double(image);% passe l'image en double
    image2=reshape(image2,[],3,1)-repmat(moyenne,nbp,1);
    image2=sum((covariences_inv*image2').*image2',1);% application de la formule de
mahalanobis
    maha=reshape(image2,h,l);
end
```

Suivi.m

```
clear all
close all

% Transfert video à image
video = VideoReader ('vid_in.mp4'); %
lire la vidéo
load('modele.mat'); % chargement des
paramètres
load('modeleDoigt.mat'); % chargement
des paramètres
mu=M(1,:);
S=M(2:4,:);
N = video.NumberOfFrames; % nombre
d'image dans la vidéo

%Motif à incorporer
Imotif=imread('herisson.jpg');

%Initialisation de la vidéo de sortie
aviobj=VideoWriter('video.avi','Uncomp
ressed AVI');
open(aviobj)


for n=1:N % parcourt la vidéo image
par image
    n
    % Segmentation à l'aide du modèle
    I = double(read (video, n));
    DmalaI=mahalanobis(I,mu,S);  %
calcul de la distance au modèle M pour
chaque pixel de l'image

    % Seuillage de l'image à l'aide de
Vseuil (compris entre 0 et 1)
    IMax = max(max(DmalaI)); % on
trouve la valeur max de l'image
    IGris = DmalaI./IMax;  %on divise
tout les point par cette valeur max
--> permet d'attribuer à chaque pixel
une valeur entre 0 et 1
%on obtient donc une image en niveaux
de gris

    ISeuillee = zeros(size(IGris));
%on crée une matrice vide de zéro
(correspond à des pixels blancs)
    ISeuillee(IGris < Vseuil) = 1;% si
la valeur du pixel < Vseuil, pixel mis
en noir

    ISeuilleeDoigt =
zeros(size(IGris));
    ISeuilleeDoigt(IGris > 0.55) = 1;

    % oprétateur morphologique
    se = strel('arbitrary',50);
    ISeuillee = imerode(ISeuillee,se);
    ISeuillee =
imdilate(ISeuillee,se);

    % Labellisation
    [imageLabellisee,nbrzone] =
bwlabel(ISeuillee);

    %Calcul des barycentres des
différentes zone de l'image labellisée

barycentres=bary(imageLabellisee,nbrzo
ne);

%     %test : affiche les 4
barycentres
%     imshow(uint8(I));
%     hold on;
%
plot(barycentres(1,1),barycentres(2,1)
,'ro');
```

```
%
plot(barycentres(1,2),barycentres(2,2)
,'ro');
%
plot(barycentres(1,3),barycentres(2,3)
,'ro');
%
plot(barycentres(1,4),barycentres(2,4)
,'ro');

    %Comparaison pour sortir une
matrice de barycentres ordonnées
   if (n==1)
        %On s'est assuré par notre
modèle d'avoir uniquement 4 zones lors
        %du traitement de la première
image

        %Ordonner les barycentres de
la 1ère image
        newBaryOrdonnes=zeros(2,4);
%on cree la matrice de barycentres
ordonnés
        cpt=0;
        for i=1:4 %On fait une boucle
qui trouvera les 4 xMin

xMin=min(barycentres(1,:)); % on
trouve le xMin
            for j=1:(4-cpt) % On
parcourt la matrice
                if
(barycentres(1,j)==xMin) %On place le
xMin a la bonne place dans BaryOrdonne

newBaryOrdonnes(1,i)=barycentres(1,j);

newBaryOrdonnes(2,i)=barycentres(2,j);
                    col=j;
                end
            end
            barycentres(:,col)=[];
            cpt=cpt+1;
        end

        if
(newBaryOrdonnes(2,1)>newBaryOrdonnes(
2,2))

ValInter=newBaryOrdonnes(:,1);%on
place la colonne dans une variable
intermédiaire

newBaryOrdonnes(:,1)=newBaryOrdonnes(:
,2);

newBaryOrdonnes(:,2)=ValInter;
        end

        if
(newBaryOrdonnes(2,3)<newBaryOrdonnes(
2,4))

ValInter=newBaryOrdonnes(:,3);%on
place la colonne dans une variable
intermédiaire

newBaryOrdonnes(:,3)=newBaryOrdonnes(:
,4);

newBaryOrdonnes(:,4)=ValInter;
        end
     % l'image sort dans le mauvais
sens
     % on change l'ordre des
barycentres

        saveNewBarryOrdonnes =
newBaryOrdonnes;
        newBaryOrdonnes(:,2) =
saveNewBarryOrdonnes(:,4);
        newBaryOrdonnes(:,4) =
saveNewBarryOrdonnes(:,2);


    else %on ne peut pas comparer la
1ère image
      %Comparaison de deux images


[newBaryOrdonnes]=distance(oldB,baryce
ntres);

    end

    %Incrustation
    Ivideo = read (video, n);

    x=zeros(1,4);
    y=zeros(1,4);
    for k=1:4
        x(1,k)=newBaryOrdonnes(1,k) ;
        y(1,k)=newBaryOrdonnes(2,k);
    end

    scale=0.80;

    mask=zeros(size(Ivideo));
     %imshow(Ivideo);
     if (i == 1)
                videoR =
uint8(zeros(size(frame, 1),size(frame,
2), 3, N));
      end

    frame =
motif2frame(Imotif,Ivideo,x,y,scale,IS
euilleeDoigt);
    writeVideo(aviobj, frame);
    oldB=newBaryOrdonnes; %on
enregistre le barycentre de l'image
qui va devenir l'"image précédente"

end

close(aviobj);
```

Bary.m

```
function [barycentres] = bary(L,nbrzone) %Image seuillée
%prend le nombre de zone et l'image labellisée et renvoie les barycentres
%des zones labellisées

barycentres=ones(2,nbrzone); %On crée un tableau vide pour rentrer la valeur des
barycentres

    for i=1:nbrzone
        [lig,col]=find(L==i);
        coord_y=mean(lig); % en repère géométrique la coordonées en y est en repère
matricielle la coordonnée de lignes
        coord_x=mean(col); % le x géométrique correspond aux colonnes
        barycentres(1,i)=(coord_x);
        barycentres(2,i)=(coord_y);
    end
end
```

distance.m

```
function [dist] = distance(oldB,newB)

    %Création tableau de distance entre les barycentres
     [a,b] = size(oldB);
     nbBold=b; %nombre de barycentres de l'image précédente
     [c,d] = size(newB);
     nbBnew=d;
     dist=ones(nbBnew,nbBold); %création tableau avec le nombre de barycentres de
l'image actuelle en abscisses
                               %et le nombre de barycentres de l'ancienne image en
ordonnées
     for i=1:nbBnew
         for j=1:nbBold
             dist(i,j)=sqrt((newB(1,i)-oldB(1,j)).^2+(newB(2,i)-oldB(2,j)).^2);
         end
     end

     dist=TrouveBary(dist,newB);
end
```

TrouveBary.m

```
function [bary] = TrouveBary(distance,newB)
    [nbligne,nbcolonne]=size(distance);
    min=distance(1,1);
    Xmin=1;
    Ymin=1;
    coordonnee=zeros(2,4);
    for n=1:4
        for i=1:nbligne
            for j=1:nbcolonne
                if(min==-2)
                    min=1000000;
                end
                if(distance(i,j)~=-1)
                    if (distance(i,j)<min)
                        min=distance(i,j);
                        Xmin=i;
                        Ymin=j;
                    end

                end
            end
        end
        coordonnee(1,Ymin)=newB(1,Xmin);
        coordonnee(2,Ymin)=newB(2,Xmin);
        for i=1:nbligne
            distance(i,Ymin)=-1;
        end

        for j=1:nbcolonne
            distance(Xmin,j)=-1;
        end
        min=-2;
    end
    bary=coordonnee;
end
```

motif2frame.m

```
function frame=motif2frame(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
```

Initialisation_doigt.m

```
%CREATION DU MODELE DU DOITGH
clear all
close all

% Transfert video à image
video = VideoReader ('vid_in.mp4'); %
lire la vidéo
n = 44;
I = double(read (video, n));
imshow(uint8(I)) % affiche l'image

% Selection zone
zoom on; % active le zoom
pause(); % enleve le zoom des qu'une
touche est appuyée
nb_points = 2;
[x,y] = ginput (nb_points); % permet
de trouver les abscisses et ordonnées
des 2 points cliqués

selection = [x(1),y(1),x(2)-x(1),y(2)-
y(1)]; % correspond à x0 y0 (point bas
gauche) longueur largeur
rectangle('Position', selection
)%tracer le rectangle
IS = double(imcrop (I,selection)); %
extraction de la sélection
figure(2), imshow(uint8(IS))

% Calcul modele
    % calcul de la moyenne mu
    ISR = IS(:,:,1); % matrice de la
composante rouge de IS
    R = ISR(:); % ISR sous forme de
vecteur colonne
    muR = mean(R); % moyenne de la
composante rouge de IS
    ISV = IS(:,:,2); % matrice de la
composante verte de IS
    V = ISV(:);
    muV = mean(V);
    ISB = IS(:,:,3); % matrice de la
composante bleue de IS
    B = ISB(:);
    muB = mean(B);
    mu = [muR muV muB];

    % calcul de la matrice de
covariance Sigma
    S11 = sum((R-muR).*(R-muR)); % R -
muR correspond à un vecteur colonne
contenant (xiR - muR)²
    S12 = sum((R-muR).*(V-muV));
    S13 = sum((R-muR).*(B-muB));
    S21 = S12; % car les termes
antidiagonaux sont égaux
    S22 = sum((V-muV).*(V-muV));
    S23 = sum((V-muV).*(B-muB));
    S31 = S13;
    S32 = S23;
    S33 = sum((B-muB).*(B-muB));

    S = [S11 S12 S13; S21 S22 S23;
S31 S32 S33];
M = [mu;S]; % le modele M

%Calcul de la distance de Mahalanobis
DmalaI=mahalanobis(I,mu,S);

%Determination du seuil
Vseuil=0.06;
[l,c,a]=size(I);
I2=ones(l,c,a);%Création d'une image
vide
for i=1:l
        for j=1:c
            if (DmalaI(i,j)<Vseuil)
                I2(i,j,1)=255;
                I2(i,j,2)=0;
                I2(i,j,3)=0;
            else
                I2(i,j,1)=0;
                I2(i,j,2)=0;
                I2(i,j,3)=255;
            end
        end
end
imshow(uint8(I2));

%Sauvegarde le modèle dans le fichier
destination
save('modeleDoigt.mat', 'mu', 'S');
```