Rapport de projet de réalité augmentée :

Au cours des mois de mars à mai, nous avons travaillé sur un projet de réalité augmentée par traitement d’image. La problématique du sujet consistait à remplacer dans une courte vidéo une feuille vierge bleu clair, dont les quatre coins étaient visualisés par 4 picots bleu foncé par une image de notre choix. Pour cela, nous avons utilisé Matlab, créant nos propres fonctions ainsi qu’utilisant celles proposées par le logiciel. Nous allons au cours de ce rapport décrire notre processus de réflexion au cours de la réalisation de ce projet, ainsi que son implémentation sous Matlab.

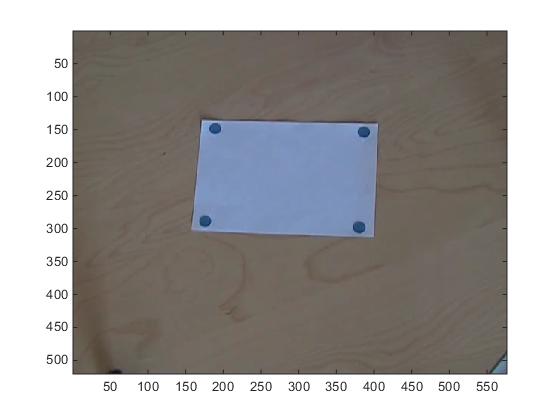
**Démarche algorithmique :**

Afin d’atteindre notre objectif de remplacer la feuille par une image, l’algorithme général peut-être décomposé en 4 étapes principales :

* Phase d’initialisation
* Repérage de la position des picots
* Incrustation de l’image dans la vidéo
* Prise en compte des doigts de la main tenant la feuille dans l’incrustation

Nous allons maintenant détailler ces étapes :

La vidéo fournie pour le projet représente une main faisant glisser une feuille bleue sur une table, les 4 coins de la feuille étant délimités par des points plus sombres :



Première image de la vidéo

Afin de pouvoir remplacer la feuille par une image différente, il faut pouvoir détecter les pixels de l’image correspondant à la feuille. Mais déterminer tous ces pixels par la couleur ne donnerait pas des bords de feuille droits, il est plus simple de détecter les 4 picots bleu foncé et d’en déduire la feuille. Ces picots sont détectés par leur couleur. La phase d’initialisation permet de détecter la couleur des picots dès la première image de la vidéo, afin de ne plus avoir à effectuer cette démarche pour le reste des images (ce qui serait très couteux en temps). Il s’agit de sélectionner sur cette première image la zone de la couleur nous intéressant, ici l’intérieur d’un picot, puis d’en déduire le modèle colorimétrique de cette zone et enfin la distance en couleur de chaque pixel de l’image à la couleur sélectionnée (distance de Mahalanobis).

Une fois l’initialisation faite, le modèle colorimétrique reste valable pour toute les images de la vidéo ; nous pouvons alors détecter la position des picots pour chacune de ces images. Pour cela, il faut d’abord effectuer un seuillage des images de distance de Mahalanobis. Ce seuillage nous fournit alors une image noire avec pour seules zones blanches les picots. Par traitement nous obtenons les coordonnées des barycentres des picots, qui sont ensuite ordonnancés afin qu’un picot pour une image corresponde au même picot dans l’image suivante.

Une fois les positions des 4 picots connues, il est désormais possible d’incruster notre image sur la vidéo par homographie : les coins de notre image sont placés aux picots. C’est ici que l’ordonnancement des barycentres des picots rentre en jeu : il est essentiel pour que d’une image à l’autre dans la vidéo, notre image rajoutée ne change pas de sens. On agrandit ensuite l’image rajoutée afin qu’elle remplace toute la feuille (les picots ne peuvent pas être placés aux coins exacts de la feuille).

Il reste cependant encore un problème : notre incrustation a remplacé tous les pixels de la feuille par ceux de notre image ajoutée, mais elle a aussi remplacé les pixels correspondant à la main tenant la feuille. Si nous voulons que la vidéo soit réaliste, il faut faire en sorte que la main soit visible par-dessus notre image rajoutée. Pour cela, il faut d’abord détecter quels sont les pixels correspondants à la main sur une image. On utilise une méthode similaire à celle employée pour détecter les picots : on détecte la couleur de la main, on image la distance de Mahalanobis à cette couleur pour l’ensemble de l’image, puis on seuille. Les pixels non nuls restants correspondent à la main, on peut alors les prendre en compte lors de l’incrustation.

**Implémentation :**

Après avoir expliqué le principe général de chacune des étapes de l’algorithme, détaillons maintenant le processus d’implémentation de cet algorithme sous Matlab.

Au cours de la phase d’initialisation, on travaille sur la première image de la vidéo. Il s’agit d’effectuer tous les traitements nécessaires sur la première image ; ces traitements serviront alors de référence pour le traitement de toutes les autres images de la vidéo. Sur cette image, on voit la feuille avec ses 4 picots, sans la main. On commence tout d’abord par sélectionner la zone de pixels de la couleur qui nous intéresse (ici le bleu foncé). Cette sélection peut se faire manuellement ou automatiquement. La zone devient alors une image (très) réduite de notre image de départ.On applique alors à cette image une fonction permettant de déterminer la matrice M correspondant au modèle colorimétrique de notre image réduite (Cette fonction sera réutilisée par la suite pour la détection de la main). La matrice M permet ensuite de calculer la distance de Mahalanobis des chacun des pixels de la première image. On a en effet :

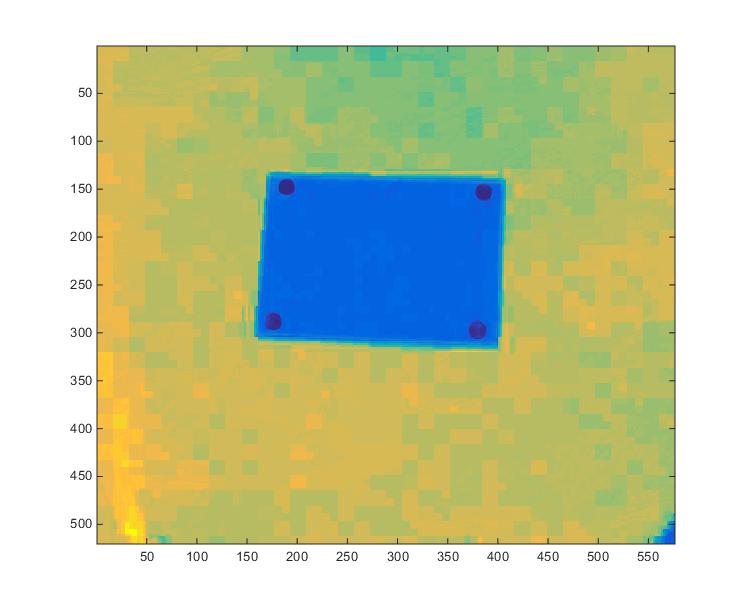
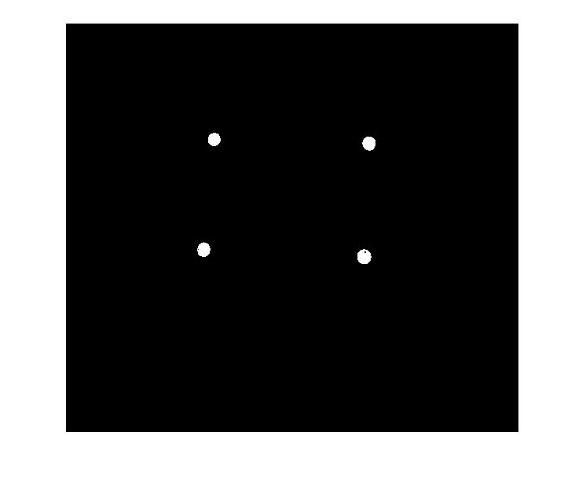
Où y est la valeur (vecteur de 3 éléments : R, G, B) du pixel et µ la moyenne des valeurs. La distance calculée est initialement présentée sous forme de matrice, qui est réorganisée afin d’obtenir une image :

Image des distances de Mahalanobis

 On cherche ensuite à appliquer un seuil à cette image, afin de n’avoir que les picots à 1 et tous les autres pixels valant 0. Le seuil choisi pendant l’initialisation sera celui utilisé pour toutes les autres images :

On détecte ensuite le barycentre de chacun des picots. Pour cela, on effectue une transformation morphologique de l’image (grâce à la fonction bwmorph). Ici, la transformation choisie fixe le pixel à si 5 ou plus de ses voisins valent 1, sinon il est fixé à 0. Puis, à l’aide de la fonction edge, nous sélectionnons seulement le contour de chaque tache présente sur l’image. Il est alors facile de déterminer le barycentre de chacune des taches. Les 4 paires de coordonnées sont alors stockées dans une matrice.

La phase d’initialisation nous permet ainsi d’obtenir la matrice M de la couleur des picots, la valeur du seuil à appliquer ainsi que les coordonnées des barycentres pour la première image.

Une fois ce traitement préalable réalisé, on effectue le même traitement pour chaque image de la vidéo, l’une après l’autre : On détermine l’image des distances de Mahalanobis (en utilisant la matrice M), on seuille, puis on détermine la position des barycentres de picots. Cependant, ces barycentres étant classés dans la matrice dans un ordre quelconque, il faut les ordonnancer afin de pouvoir réaliser facilement l’homographie par la suite. Pour cela, on crée une matrice contenant les distances entre les barycentres de l’image et les barycentres de l’image précédente, et on ne garde que les coordonnées correspondant aux 4 distances les plus petites.

Ainsi, l’étape de repérage des positions des picots nous permet d’avoir en sortie de cette étape la position des barycentres des picots, stockés dans une matrice.

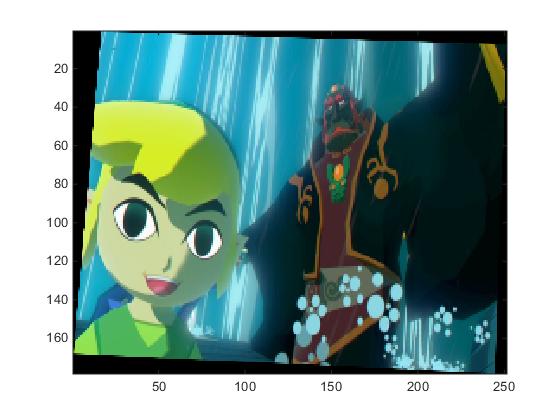
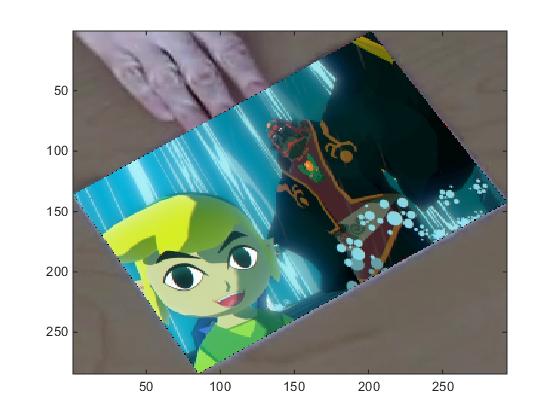
Une fois ces positions ordonnancées pour une image déterminées, nous pouvons passer à l’incrustation de notre image à l’image de départ. Pour cela, on réalise une homographie grâce aux fonctions Matlab fitgeotrans et imwarp. La fonction fitgeotrans prend en entrée les cordonnées des coins de l’image que l’on veut transformer (ici notre image à ajouter) et les cordonnées des coins de la feuille et calcule quelle est la transformation exacte à appliquer à notre image afin que celle-ci ait les mêmes dimensions que la feuille. Puis la fonction imwarp applique à notre image la transformation déterminée précédemment. (Remarque : les coordonnées des barycentres des picots ne correspondant pas aux coins exacts de la feuille, les coordonnées rentrées dans fitgeotrans sont légèrement modifiées afin de correspondre aux coins de la feuille.)

Image après transformation

Il s’agit ensuite de remplacer les pixels de l’image correspondant à la feuille par ceux de notre image de remplacement.



Remplacement dans l'image

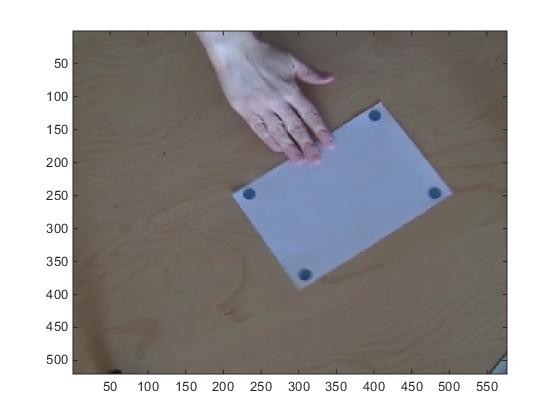
Cependant, cette opération incruste notre image de remplacement sur l’ensemble de la feuille ainsi que sur une partie de la main tenant la feuille. Il faut donc exclure la main de l’ensemble des pixels à ressembler. Pour cela, il faut détecter, par la couleur, les pixels correspondant à la main. On effectue donc une initialisation similaire à l’initialisation sur la couleur des picots, cette fois ci en travaillant non pas avec la première image de la vidéo, mais avec la première vidéo où apparait la main :

Image d'initialisation pour la main

On détermine la matrice de modèle colorimétrique pour la main, ainsi qu’une valeur de seuil adaptée. Ensuite, lors de l’incrustation, on calcule l’image des distances de Mahalanobis pour la main, et on seuille cette image, puis on lui applique une fermeture et une ouverture afin d’homogénéiser la surface de la main obtenue. Les pixels valant 1 sur cette image (qui correspondent aux pixels de la main) sont alors exclus des pixels à remplacer (on leur attribue la valeur -1 lors du remplacement pour qu’ils ne soient pas pris en compte).

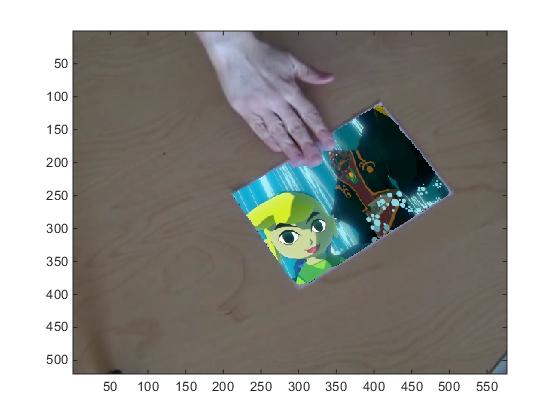
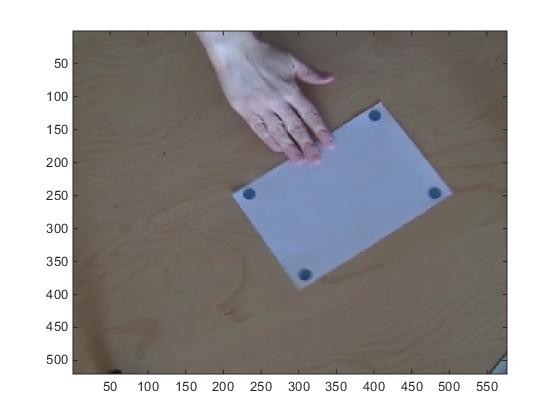
**Résultats :**

Le programme final prend donc en entrée une vidéo montrant la feuille avec ses quatre picots bleus, ainsi que l’image servant pour le remplacement. Il y a également deux paramètres, le nom du fichier pour la vidéo de sortie, après remplacement ; et un paramètre ‘inception’ qui détermine le nombre de fois que le traitement doit être effectué sur une même image (l’image de remplacement devient la vidéo elle-même).

On obtient donc plusieurs vidéos possibles en sortie :

* Sortie1.avi correspond à un traitement simple. Le traitement de la présence des doigts a été optimisé, mais on observe quand même certains problèmes de couleur.
* Sortie2.avi à un traitement ‘inception=2’. Le traitement est plus long pour obtenir ce résultat, car aucune optimisation n’a été faite pour ce traitement. On pourrait améliorer grandement le temps de traitement en diminuant la résolution des premiers traitements (images intérieures, très petites sur l’image finale)
* Sortie3.avi correspond à un traitement dans lequel la couleur de la main a été mal choisie.

Image de départ et image finale



**Conclusion :**

Au terme de cette étude, nous avons donc, par le biais de quatre étapes principales, pu remplacer sur une vidéo une feuille par une image de notre choix. Bien que le code soit relativement long, son exécution pour une courte vidéo comme celle que nous avons utilisé n’est que de quelques minutes. Cependant, il est probable que les fonctions prédéfinies de Matlab pourraient être optimisées pour notre utilisation spécifique, ce qui nous ferait gagner en temps d’exécution.

**Code :**

clear, tic

%PARAMETRE :

VideoALire = 'vid\_in.mp4'; %Vidéo sur laquelle se fait le traitement

ImgRemplacement='img\_remplacement.jpg'; %Image utilisée pour le remplacement.

VideoSortie='sortie4.avi'; %Fichier de sortie pour la vidéo traitée

inception = 1; %Nombre de fois que le traitement doit se réaliser par image.

%Mettre 1 pour un traitement normal.

%-------------Début du programme-----------------

%Chargement des fonctions du dossier 'fonctions'

addpath('fonctions')

%Définie la valeur globale N, numéro de l'image en cours de traitement

global N

%Récupération des information concernant la vidéo sur laquelle se fait le traitement :

Video = VideoReader(VideoALire);

vidHeight = Video.Height;

vidWidth = Video.Width;

NumberOfFrames = Video.NumberOfFrames;

%Après NumberOfFrames, la vidéo à été lue jusqu'à la dernière image. Il

%faut donc la réouvrir pour repartir du début de la vidéo :

Video = VideoReader(VideoALire);

%Définition de l'image utilisée pour faire le remplacement :

ImgRemplacement = double(imread(ImgRemplacement));

%Définition du fichier de sortie :

writerObj = VideoWriter(VideoSortie);

open(writerObj);

sortieVide=zeros(vidHeight, vidWidth);

%Initialisation des variables utilisée plus tard :

barN = zeros(NumberOfFrames\*2, 4);

M2=0;

seuil=0;

%Le modèle M2 et le seuil sont utilisés pour la prise en compte de la mains.

%Il ne doivent pas être utilisés avant l'image 38 (apparition de la main)

%-------------Début du traimtement-----------------

%Traitement particulier de la première image

N=1;

img1 = readFrame(Video);

[M,s, barN(1:2,:)] = initialisation(img1);

%Boucle pour traiter l'ensemble des images de la vidéo

while(hasFrame(Video))

%Selection de l'image suivante

N=N+1;

imgN = readFrame(Video);

%Repérage de la position des picots sur cette image

barN(2\*N:2\*N+1,:) = reperage\_pos\_picots(imgN, M, s, barN(2\*(N-1):2\*(N-1)+1,:));

if N==38

[M2, seuil] = initialisationDoigts( imgN );

end

% Incrustation de l'image de remplacement

imgN = Incrustation(imgN, ImgRemplacement, barN(2\*N:2\*N+1,:), M2, seuil);

for i=1:inception-1

imgN = Incrustation(imgN, double(imgN), barN(2\*N:2\*N+1,:), M2, seuil);

end

% Enregistrement de l'image avec remplacement

writeVideo(writerObj, imgN)

if mod(N, 10)==0

disp(['Avancement : ', num2str(floor(N/NumberOfFrames\*100)), '%']);

end

end

%Finalise

close(writerObj);

close all;

toc;

msgbox('Terminé');

--------------------------------------------------------

function [ M ] = CalcM( imgRed )

%Permet le calcul du modèle M=[moyenne,covariance] des couleurs de l'image

%passée en entrée

%ENTREE : imgRed image de double en couleur

%SORTIE : M modèle des couleur, définie selon M=[mu,sigma]:

% mu (3\*1 double) moyenne des couleur sur l'espace RGB

% sigma (3\*3 double) covariance des couleur sur l'espace RGB

mu = mean(mean(imgRed));

mu=mu(:); %Réordonnancement des moyenne

%Récupération et réodonancement des couleur de chaque pixel

R=imgRed(:,:,1);

G=imgRed(:,:,2);

B=imgRed(:,:,3);

x(:,1)=R(:);

x(:,2)=G(:);

x(:,3)=B(:);

%Calcul de chaque élément de la matrice de covariance

sigma=ones(3,3);

for i=1:3

for j=1:3

sigma(i,j)=sum((x(:,i)-mu(i)).\*(x(:,j)-mu(j)));

end

end

M=[mu,sigma];

end

--------------------------------------------------------

function [barycentre] = determination\_bar(im\_seuil)

%Determine le barycentre de chaque élément indépendant contenu dans l'image

%en entrée

%ENTREE : im\_seuil image en noir et blanc contenant plusieurs zone distance

% dont on cherche les barycentres

%SORTIE : barycentre (2\*n double) position [x;y] des barycentre des n zones

% distinctes trouvée

im\_seuil = bwmorph(im\_seuil, 'majority', Inf);

im\_seuil= edge(im\_seuil, 'canny', [0.1 0.5],3);

[L,num] = bwlabel(im\_seuil,8);

barycentre = zeros(2,num);

for ii=1:num

im\_seuil=(L==ii);

[y,x] = find(im\_seuil);

barycentre(1,ii) = mean(x);

barycentre(2,ii) = mean(y);

end

--------------------------------------------------------------------

function [ imDist ] = dMaha( img, M )

%Calcul la distance de Mahalanobis entre chaque pixel de l'image img en entrée

%et un modèle de couleur donné par M

%ENTREES : img image de double en couleur

% M matrice modèle de couleur retournée par CalcM

%SORTIE : imDist image de double en nuance de gris donnant la distance de

% Mahalanobis pour chaque pixel

%Séparation des moyenne et des covariances

sigma = M(:,2:4);

mu = M(:,1);

%Séparation des données pour chaque couleur

R = img(:,:,1)-mu(1);

G = img(:,:,2)-mu(2);

B = img(:,:,3)-mu(3);

%Calcul de la distance de Mahalanobis pour chaque pixel. Renvoie les pixels

%comme un unique vecteur ligne

imDist = sum([R(:)'; G(:)'; B(:)'].\*(inv(sigma)\*[R(:)'; G(:)'; B(:)']),1);

%Réorganisation des pixels sous forme d'un images

imDist = reshape( imDist, size(img,1), size(img,2), 1 );

end

--------------------------------------------------------------------

function [ imgN ] = Incrustation( imgN, imgRemplacement, barN, Mmain, seuilmain)

%Dans l'image imgN, remplace la zone définie par les 4 picots de barN par

%l'image imgRemplacement. Ne remplace pas la zone correspondant à la main

%ENTREES : imgN : Nième image (int8) de la vidéo

% imgRemplacement : image (double) servant à remplacer la zone

% définie par les picots

% barN : poistion (2\*4 int8) des barycentres des picots sur

% l'image N

% Mmain : Matrice modèle retournée par CalcM représentant les

% couleurs de la main

% seuilmain : seuil permetant d'isoler la couleur de la main

%SORTIE : imgN : Nième image (int8) de la vidéo, avec le remplacement

% effectué

%Récupère la variable globale N indiquant le numéro de l'image en cours de

%traitement

global N

%Réorganise les barycentres

barN=barN';

%Définit la zone dans d'imgN représentée par barN. corecCoin permet de ne

%pas prendre les bords de l'image de remplacement, et ainsi remplacer une

%zone plus large (ie : toute la feuille), au lieu de remplacer la zone

%située exclusivement entre les picots.

corecCoin=0.080;

coins = [(1-corecCoin)\*size(imgRemplacement,2), (1-corecCoin)\*size(imgRemplacement,1); corecCoin\*size(imgRemplacement,2), (1-corecCoin)\*size(imgRemplacement,1);

corecCoin\*size(imgRemplacement,2), corecCoin\*size(imgRemplacement,1); (1-corecCoin)\*size(imgRemplacement,2), corecCoin\*size(imgRemplacement,1)];

%Calcul de l'homographie entre la zone à remplacer sur l'image de la vidéo

%et l'image de remplacement.

H=fitgeotrans(coins, barN, 'projective');

%Déforme l'image de remplacement d'après l'homographie, récupère ses

%dimensions

[imgtransformee,infos] = imwarp(imgRemplacement, H, 'FillValues', [-1 -1 -1]);

minX = round(infos.XWorldLimits(1));

minY = round(infos.YWorldLimits(1));

%Isole la partie d'intérêt (ie : à modifier) sur l'image imgN

imgNfinale = imgN(minY:(minY+infos.ImageSize(1)-1), minX:(minX+infos.ImageSize(2)-1),:);

%Si la main est présente sur l'image, donc à prendre en compte

if N>=38

%Répère la position de la main

imDist = dMaha(double(imgNfinale), Mmain);

imSeuillee = imDist < seuilmain;

%Ferme puis ouvre l'image de la main, pour homogénéiser la zone à

%prendre en compte

elem = strel('disk', 2, 0);

imSeuillee = imdilate(imerode(imSeuillee, elem), elem);

elem2=strel('disk', 4, 0);

imSeuillee = imerode(imdilate(imSeuillee, elem2), elem2);

%Indique que la main ne doit pas être remplacée (valeur -1 sur l'image

%de remplacement)

imgtransformee = (repmat(imSeuillee==0,1,1,3)).\*imgtransformee + (repmat(imSeuillee~=0,1,1,3))\*-1;

end

%Remplie l'image de remplacement après homographie (donc non rectangulaire)

%avec les valeur de l'image de fond imgN

imgNfinale = (imgtransformee<0).\*double(imgNfinale) + (imgtransformee>=0).\*imgtransformee;

%Replace l'image après incrustation dans l'image globale

imgN(minY:(minY+infos.ImageSize(1)-1), minX:(minX+infos.ImageSize(2)-1),:) = imgNfinale;

end

--------------------------------------------------------------------

function [M, seuil, barN\_1] = initialisation(img1)

%Cette fonction permet l'initialisation du traitement de la vidéo.

%Elle permet de selectionner l'un des picots bleu de déterminer un seuil

%adéquat qui sera utiliser pour tout le traitement

%ENTREE : img1, 1ière image de la vidéo

%SORTIES : M, matrice (3\*4 double) modèle utilisée par la distance de mahalanobis

% seuil, seuil (double) permetant de diférencier les picots du reste

% barN\_1, position (2\*4 double) des barycentre des picots de la

% première image. Chaque colonne représente les position [x;y] de

% chaque barycentre.

close all

imshow(img1);

%Selection d'un picot à la souris :

% [x,y] = ginput(2);

% x=fix(x)

% y=fix(y)

%Les valeurs suivante fonctionnent bien :

x=[184,194];

y=[142,152];

%Reduction de l'image d'après les coordonnées choisies

imgRed = img1(y(1):y(2),x(1):x(2),:);

%Calcul du modèle M utilisé pour faire une distance de Mahalanobis

M=CalcM(double(imgRed));

%Cacul de la distance de Mahalanobis

imDist = dMaha(double(img1), M);

% Choix d'un seuil pour différencier la couleur du picot du reste

seuil = '1';

%Boucle permettant de trouver un seuil adéquat ne faisant apparaitre que

%les picots. Cette boucle peut être commentée pour utiliser le seuil apr

%défaut de 1

validation = 'Non';

imagesc(imDist);

while validation == 'Non'

seuil = inputdlg('Veuillez choisir une valeur pour le seuil',...

'Choix seuil',...

1,...

{seuil});

seuil = str2double(seuil);

imSeuillee = imDist < seuil;

h=figure, imshow(imSeuillee);

validation = questdlg('Ce seuil vous convient-il ?', 'Validation seuil',...

'Oui', 'Non', 'Oui');

seuil = num2str(seuil);

close all

end

seuil = str2double(seuil);

%Seuillage

imSeuillee = imDist < seuil;

%Détermination de la position des picots

barN\_1 = determination\_bar(imSeuillee);

%Réordonancement des picots trouvés

barry = barN\_1(:,3);

barN\_1(:,3) = barN\_1(:,4);

barN\_1(:,4) = barry;

close all

end

--------------------------------------------------------

function [ M, seuil ] = initialisationDoigts( imgDoigts )

%Initialise la prise en compte de la présence des doigts sur l'image :

%récupère un matrice modèle M et un seuil permettant d'isoler la couleur

%des doigts

%ENTREE : imgDoigts : image (int8) de la video d'origine dans laquelle les

% doigts sont présents

%SORTIE : M : matrice modèle retournée par CalcM permettant d'isoler la

% couleur des doigts

% seuil : valeur (double) permettant d'isoler la couleur des doigts

imagesc(imgDoigts);

%Selection des doigts à la souris :

[x,y] = ginput(2);

x=fix(x)

y=fix(y)

% Les valeurs suivante fonctionnent bien :

% x=[277, 286];

% y=[110, 130];

%Reduit l'image à la zone où se trouvent les doigts

imgRed = imgDoigts(y(1):y(2),x(1):x(2),:);

M=CalcM(double(imgRed));

seuil = 0.1;

%La valeur de seuil fonctionnelle est empirique

end

function [ bar ] = ordonnancement( bar1, bar2 )

%Ordonne les 4 barycentre de bar2 d'après leur proxymité avec les

%barycentre de bar1

%ENTREES : bar1 : (2\*4 int8) barycentre de reférence

% bar2 : (2\*4 int8) barycentre à ordonnée

%SORTIE : bar : (2\*4 int8) barycentre de bar2 réordonnés (colonnes

% échangées)

[size1, size2] = size(bar2);

dist = zeros(4,size2);

bar = zeros(2,4);

for i = 1:4

for j=1:size2

dist(i,j) = pdist([bar1(1,i),bar1(2,i);bar2(1,j), bar2(2,j)], 'euclidean');

end

end

for i = 1:4

[m, pos\_old] = min(dist);

[mini, pos\_new] = min(m);

size(bar2);

bar(:, pos\_old(pos\_new)) = bar2(:, pos\_new);

dist(pos\_old(pos\_new),:) =max(max(dist))+3;

dist(:,pos\_new) = max(max(dist))+3;

end

end

--------------------------------------------------------------------

function [ barN ] = reperage\_pos\_picots( imgN, M, seuil, barN\_1 )

%Repère la position (barycentre) des 4 picots de la Nième image d'après la

%postion des picots à l'image précédente

%ENTREES : imgN : Nième image (valeur en int8) de la vidéo

% M : matrice modèle utilisée pour la distance de Mahalanobis.

% Obtenue à l'aide de CalcM

% Seuil : double, seuil à prendre pour isoler le picots dans

% l'espace de Mahalanobis

% barN\_1 : position (2\*4 int8) des 4 barycentres à l'image précédente

%SORTIE : barN : position (2\*8 int8) des 4 barycentres à l'image N

%Crée une image binaire isolant les 4 picots

imgN = double(imgN);

imDist = dMaha(imgN, M);

imSeuil = imDist < seuil;

%Détermines les barycentre des 4 picots

barycentre = determination\_bar(imSeuil);

%Réordonne les picots d'après leur position à l'image précédente

barN = ordonnancement(barN\_1, barycentre);

end