
# [ PROJET IMAGE : RECALAGE PAR CORRELATION DE PHASE ]

[ TS225 ]

Brizard Mattis & Reclus Alexis
mbrizard001@enseirb-matmeca.fr / areclus@enseirb-matmeca.fr

# Table des matières

# I. Introduction

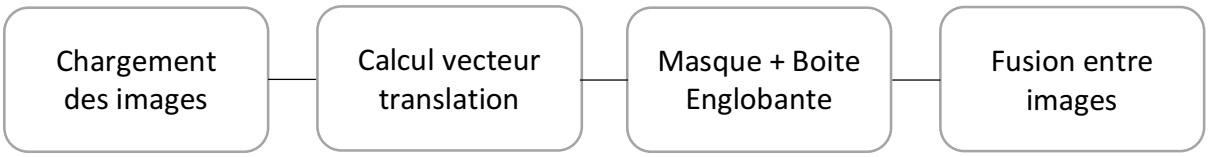
Dans ce projet, nous essayons de concevoir un programme permettant de construire une mosaïque d'images à partir de plusieurs images initiales représentant chacune une partie d'une même scène.

La méthode que nous développons doit répondre à deux critères bien précis qui sont :

- Le recalage doit être entièrement automatique ;
- Le temps de calcul doit être inférieur à 1 seconde par image à fusionner.

Le projet s'est décomposé en plusieurs étapes, tout d'abord nous avons réalisé un programme permettant de retrouver la position d'une image par rapport à une autre image. Puis un autre programme afin de créer une image résultante contenant les deux images replacées et fusionnées.

Pour réaliser la fusion, nous effectuons les étapes suivantes :




Pour N images, nous répétons ce processus N-1 fois.

Dans ce rapport, nous détaillerons d'abord les parties théoriques, telles que la corrélation de phase et la construction de la mosaïque. Ensuite nous parlerons de notre algorithme, des choix que nous avons fait, des résultats ainsi que des performances obtenus. Pour terminer, nous expliquerons notre organisation et le temps que chacun d'entre nous a passé sur les tâches du projet.

# II. Partie Théorique

## 1. Corrélation de phase

Pour retrouver la position relative d'une image par rapport à une autre image, nous utilisons la méthode de translation par corrélation de phase. Pour ce faire, nous appliquons aux deux images la transformée de Fourier :

$$F(u,v) = \iint f(x,y)e^{-j2\pi(xu+yv)}dxdy$$

On calcule ensuite le spectre de puissance croisée $P(u,v)$ des deux images avec la relation :

$$P(u,v) = \frac{F_2(u,v)F_1^*(u,v)}{F_2(u,v)F_1^*(u,v)} = e^{j2\pi(ux_0+vy_0)}$$

Puis en appliquant la transformée de Fourier inverse de la puissance croisée, on obtient une impulsion de Dirac dans l'image positionnée en $(x_0, y_0)$ qui correspond au vecteur translation entre les images.

## 2. Construction de la mosaïque

La première étape dans la construction de la mosaïque d'images est de créer la « boîte englobante » de chaque image à fusionner. La boîte englobante contient la position relative de l'image par rapport à une origine fixée arbitrairement. Pour ce projet nous avons choisi les coordonnées (-1000 ; -1000). La première image de taille [w x h] aura une boîte englobante ayant pour coordonnées $\begin{bmatrix} -1000 & -1000+w \\ -1000 & -1000+h \end{bmatrix}$. Pour une image quelconque, on calcule sa position relative par rapport à l'image précédente puis on crée la boîte englobante résultante correspondante de coordonnées $\begin{bmatrix} x_{min} & x_{\max} \\ y_{min} & y_{max} \end{bmatrix}$.

Conjointement à la boîte englobante, nous créons un masque résultant, comme nous travaillons sous Matlab, nous avons choisi de réaliser un masque non binaire. L'intérêt de ce masque est de pouvoir « compter » le nombre de fois où un pixel va apparaître dans l'image finale. Grâce au masque, plus un pixel sera redondant, plus il sera pris en compte pour la réalisation de l'image finale.

# III. Algorithme et Résultats

## 1. Corrélation de phase et fusion de deux images

En réalisant la méthode de la corrélation de phase, nous obtenons un spectre avec la présence du Dirac décrit précédemment.

Lorsqu'on calcule le vecteur translation entre 2 images, ce vecteur a pour origine l'origine de l'image (sur la *figure 1*, l'origine se trouvant en haut à gauche) et pour arrivée, le Dirac (point blanc) présent sur l'image de la transformée inverse du spectre de puissance.

Comme illustré sur la *figure 1*, si l'on échange les images 1 et 2, la position du Dirac ne correspond pas au vecteur translation depuis l'origine de l'image. Dans ce cas, le Dirac observé n'est en fait qu'un écho du véritable Dirac. Ce dernier se trouvant au dessus de l'origine à gauche. On observe d'ailleurs une intensité plus faible pour cet écho. Le vecteur translation entre l'image 1 et l'image 2 résultant est donc en réalité représenté par le vecteur blanc tracé sur la figure 1.

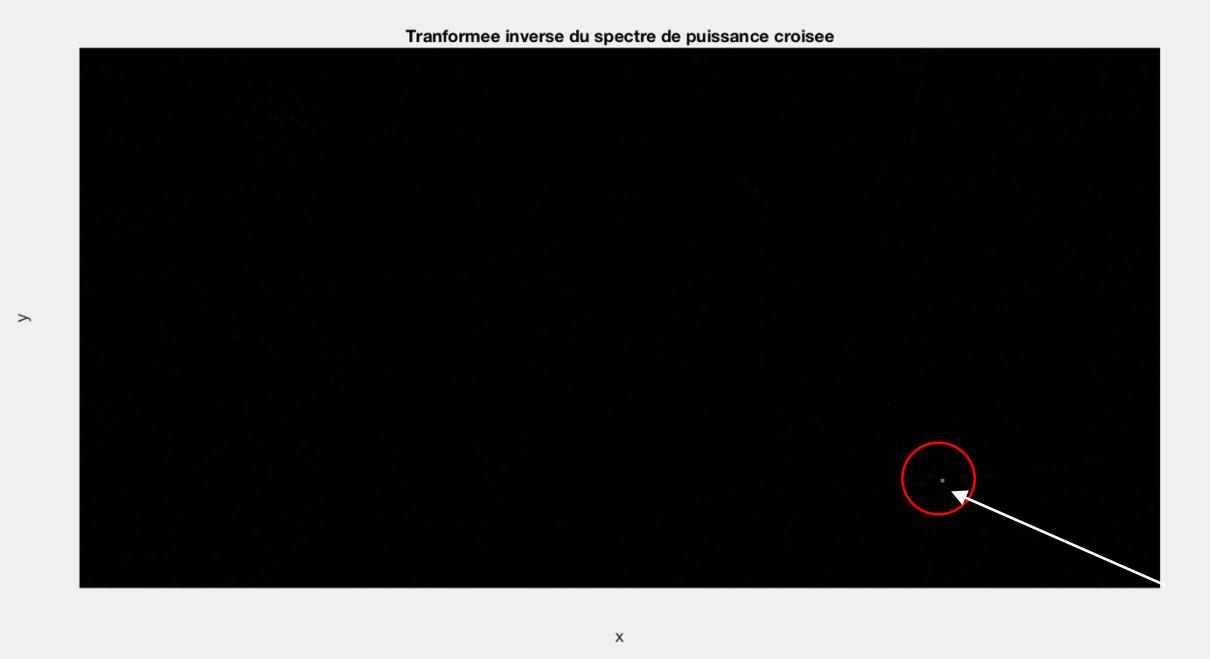



*Figure 1 – Transformée de Fourrier inverse du spectre de puissance croisée*

Nous avons fait le choix d'implémenter la méthode de fusion qui fusionne 2 images et retourne l'image fusionnée, ainsi pour 50 images, l'image 1 sera fusionnée avec l'image 2, puis l'image résultante sera fusionnée avec l'image 3 etc.

Lorsque l'on change l'ordre des images, la fonction fusion fusionne quand même les images, à condition d'avoir une partie des images en commun. Si l'on fusionne deux images n'ayant aucune partie commune, alors la corrélation de phase ne va pas trouver le vecteur translation entre les images.

Pour palier à ce problème, une fonction peut être implémentée, comme calculer les corrélations de phase entre images pour toutes les images et ainsi remettre les images « dans l'ordre » en choisissant l'image ayant la corrélation maximum pour image suivante, avant de réaliser la fusion.

### 2. Mosaïque récursive

Le but du projet étant de créer une mosaïque d'images, nous avons dû réaliser une boucle qui parcourt toutes les images fournies et les fusionne deux à deux. L'ordre dans lequel les images sont fournies n'influe pas sur l'image finale, cependant il faut que les images comparées aient une partie commune, on ne peut pas donner deux images n'ayant aucune partie en commun.

Lorsqu'on réalise le test avec le jeu d'images fournies par l'encadrant de ce projet, nous fusionnons 50 images et obtenons la mosaïque visible en *figure 2*.


*Figure 2 –Mosaïque d'images à partir de 50 images*

La figure 2 est donc l'image finale obtenue à partir des 50 images fusionnées. Nous pouvons remarquer que les images sont en noir et blanc. L'algorithme fonctionne aussi pour des images en couleur, mais une partie du code est à décommenter, il s'agit du passage en YCbCr dans le fichier *vect_trans.m*, en effet le passage en YCbCr pour des images en couleur est nécessaire pour des calculs comme la corrélation de phase par exemple.

## 3. Performances

Sur nos machines, l'algorithme met 11 secondes environ pour fusionner 50 images, il répond donc au critère de temps imposé dans le sujet du projet. Nous avons ajouté une interface de sélection qui permet à l'utilisateur de choisir les images qu'il souhaite fusionner entre elles.

Sur l'annexe 3, est présenté le code que nous avons implémenté pour tester la réalisation d'une mosaïque d'images en couleur. Cette partie nous a été utile car c'était notre démonstration avant d'implémenter l'algorithme final. Si l'on utilise le code de l'annexe 3, et que l'on décommente le passage en YCbCr dans *vect_trans.m* on obtient une mosaïque d'images comme présentée sur la figure 3.

*Figure 3 – Image fusionnée d'un phare à partir de 4 images*

Pour obtenir le résultat en figure 3, nous avons découpé une image d'origine (un phare) en 4 sous-images puis nous les avons fusionnées entre elles, comme si nous n'avions aucune information sur l'image de départ.

# IV. Organisation

## 1. Les différentes tâches et la répartition du temps

Pour le projet, nous nous sommes répartis équitablement les tâches. Pour commencer, nous avons passé du temps tous les deux sur le même ordinateur afin de mettre en œuvre un premier algorithme fonctionnel mais qui réalisait seulement le calcul de la translation. Cette partie a été nécessaire afin que nous partions tous deux du même code et que nous comprenions le sujet de la même manière. Ce qui nous a permis par la suite de toujours aller vers un objectif commun. Lorsque que nous étions sur la même machine, nous avons codé tous les deux les fonctions *boite_masques.m, trans_boite.m, vect_trans.m, correl_phase.m* et *decoupe_img.m* sur lesquels nous avons passé 5h00 environ (2 séances).

Après avoir réalisé le calcul de la translation entre les images, nous nous sommes séparés le travail pour être plus efficace. Mattis Brizard a réalisé le code permettant de charger les 50 images et de réaliser la fusion de ces 50 images (les algorithmes *mainfusion.m* et *nfusions.m*) sur lesquels il a passé 5h30 environ (2 séances). Alexis Reclus, quant à lui, a passé environ 5h30 sur l'algorithme *fusion.m,* la fonction qui réalise la fusion entre les images. Nous avons donc tous les deux passés un peu moins de 11 heures sur le projet.

## 2. Les difficultés rencontrées

Bien que le projet ait été mené à son terme, nous avons tout de même rencontré des difficultés qui nous ont amenées à réfléchir ensemble. Nous pensons que la difficulté principale a été de récupérer les positions des deux images dans l'image résultante pour les placer au bon endroit puis de créer en même temps les masques. En effet, lors de la création des masques, la première idée fût un masque binaire, 1 si le pixel était présent dans l'image finale, et 0 si ça n'était pas le cas. Cependant des erreurs se produisaient lorsque le pixel apparaissait dans différentes images. Ainsi un masque non binaire a été mis en place, une autre difficulté était que certaines valeurs assignées étaient des 0 et donc en divisant l'image par le masque pour régler l'intensité, nous nous retrouvions avec des valeurs « Not a number » dans l'image finale.

# V. Conclusion

Pendant ce travail, nous avons appris à gérer un projet en binôme, à nous organiser ensemble pour arriver à créer des algorithmes complémentaires. L'objectif du projet était de créer un algorithme permettant la création d'une mosaïque à partir d'images en respectant deux critères qui étaient un critère de temps et un critère de performance. Ces derniers ont été respectés car l'algorithme est suffisamment rapide et la reconstruction est automatisée. L'algorithme créé peut être étendu notamment dans les smartphones, lorsque l'on prend une photographie en mode « panorama ».

La méthode de corrélation de phase est surtout utilisée pour la conversion d'un standard de télévision en un autre.

Un des avantages majeurs de la méthode est l'adaptabilité dans le cas d'une rotation, il faut passer les coordonnées en polaire et choisir une grille d'échantillonnage commune.

Ainsi, les résultats obtenus rendent concrètes les applications de la reconstruction d'images par recalage de phase et nous permettent également de comprendre la réalisation de panorama en photographie.

# VI. Bibliographie


- https://en.wikipedia.org/wiki/Phase_correlation
- Documentation Matlab
- http://wolfestar.net/theriot/docs/primary/WilsonIEEETIP06.pdf
- https://www.guide-photo-panoramique.com/comment-faire-photo-panoramique.html
- http://donias.vvv.enseirb-matmeca.fr/ts225.html

# ANNEXE 1

## *mainfusion.m*

```
clear;
close all;
clc;

%% Choix des images
[filename,pathname]=uigetfile('*.bmp','multiselect','on');
for f=1:length(filename)
    images(:,:,f) =
double(imread(fullfile(pathname,filename{f})));
end

%% Appel de la fonction
ImgFinale = nfusions(images);

%% Affichage de l image fusionnee
figure
imshow(uint8(ImgFinale));

%% Calcul de l erreur en changeant l ordre
%
%
[filename,pathname]=uigetfile('*.bmp','multiselect','on');
% for f=1:length(filename)
%     images2(:,:,f) =
double(imread(fullfile(pathname,filename{f})));
% end
%
% ImgFinale2 = nfusions(images2);
%
% figure
% imshow(uint8(ImgFinale2));

%% Attention, prendre le meme nbre d images pour avoir une
somme realisable
%error = sum(sum(abs(ImgFinale2 - ImgFinale)));
```

## *boite_masques.m*

```
function [ Masque, Boite ] = boite_masques( Image )
%boite_masques genere une boite englobante et un masque
associes a une
%image originale

[h, w, s] = size(Image);
Masque = ones (h,w);
Boite = [-1000 -1000; -1000+w -1000+h];

end
```

## *nfusions.m*

```
function [Image_finale] = nfusions(images)

%nfusions retourne l image fusionnee totale du vecteur
d images fourni

[h, w, nb] = size(images);
tx = zeros(1,nb-1);
ty = tx;

%% Calcul du masque et de la boite pour la premiere
image
    [Masques(:,:,1), Boites(:,:,1)] =
boite_masques(images(:,:,1));

%% Calcul du vecteur translation, du masque et la boite
pour la deuxieme image
    [h, w] = size(images(:,:,2));
    [tx(1), ty(1)] =
vect_trans(images(:,:,1),images(:,:,2),w,h);
    [Masques(:,:,2), Boites(:,:,2)] =
boite_masques(images(:,:,2));

%% Calcul de la translation de la boite 2 par rapport a
la boite 1 et l'image resultante entre 1 et 2
    [Boites(:,:,2)] = trans_boite(Boites(:,:,1), tx(1),
ty(1));
    [Image_finale, Masque_final, Boite_finale] =
fusion(images(:,:,1), images(:,:,2), Masques(:,:,1),
Masques(:,:,2), Boites(:,:,1), Boites(:,:,2));

%% Boucle pour realiser le processus sur les n images
fournies
for i=3:nb

    %% Calcul de la taille de l image qu on fusionne
    [h, w] = size(images(:,:,i));

    %% Calcul du vecteur translation entre l image a
fusionner et l image precedente
    [tx(i-1), ty(i-1)] = vect_trans(images(:,:,i-
1),images(:,:,i),w,h);

    %% Calcul du masque et de la boite de l image a
fusionner
    [Masques(:,:,i), Boites(:,:,i)] =
boite_masques(images(:,:,i));

    %% Calcul de la tranlation de la boite de l image a
fusionner
    [Boites(:,:,i)] = trans_boite(Boites(:,:,i-1),
tx(i-1), ty(i-1));

    %% FUSION entre images
    [Image_finale, Masque_final, Boite_finale] =
fusion(Image_finale, images(:,:,i), Masque_final,
Masques(:,:,i), Boite_finale, Boites(:,:,i));

end
```

# ANNEXE 2

## *vect_trans.m*

```
function [ res_tx, res_ty ] = vect_trans( img1, img2,w, h
)
%This functions computes the translation vector between
the two images

%% Cette partie est a decommenter dans le cas ou la
mosaique d image est en couleur
% R1=img1(:,:,1);
% G1=img1(:,:,2);
% B1=img1(:,:,3);
%
% Y1=0.299*R1+0.587*G1+0.114*B1;
%
% R2=img2(:,:,1);
% G2=img2(:,:,2);
% B2=img2(:,:,3);
%
% Y2=0.299*R2+0.587*G2+0.114*B2;

%% Pour une image en couleur, decommenter la partie
precedente et changer img1 par Y1 et img2 par Y2
fft_img1 = fft2(img1);
fft_img2 = fft2(img2);

P =
fft_img2.*conj(fft_img1)./(abs(fft_img2.*conj(fft_img1)));

p = ifft2(P);

[res_ty, res_tx] = find(p == max(p(:)));

if (res_tx < floor(w/2))
    res_tx=-res_tx+1;
else
    res_tx = w-res_tx+1;

end
if (res_ty < floor(h/2))
        res_ty = -res_ty+1;
    else
        res_ty = h-res_ty+1;
end
end
```

## *trans_boite.m*

```
function [ Boite_out ] = trans_boite( Boite, tx, ty )
%trans_boite realise la translation d une boite

Boite_out(:,1) = Boite(:,1) + tx;
Boite_out(:,2) = Boite(:,2) + ty;

end
```

## *fusion.m*

```
function [ Im, M, B ] = fusion( Im1, Im2, M1, M2, B1,
B2 )
% fusion effectue la fusion de deux images et retourne
une image, un masque et une boite englobante

[h1,w1,s1] = size(Im1);
[h2,w2,s2] = size(Im2);

%% Retourne une erreur si une image est en noir et
blanc et l autre et en couleur par exemple
if (s1~=s2)
    disp('ERROR SIZE OF IMAGES');
    return;
end

%% Calcul de la boite englobante totale B a partir de
B1 et B2
B(1,:) = min(B1(1,:), B2(1,:));
B(2,:) = max(B1(2,:), B2(2,:));

%% Preallocation du masque et de l image fusionnee
resultante
taillex = B(2,1) - B(1,1);
tailley = B(2,2) - B(1,2);
M = zeros(tailley, taillex);
Im = M;

%% Calcul des positions de l image 1 dans l image
fusionne resultante
pos_im1_t = [B1(1,2) - B(1,2) + 1, B1(1,1) - B(1,1) +
1];
pos_im1_b = [pos_im1_t(1,1) + h1 - 1, pos_im1_t(1,2) +
w1 - 1];
%% Calcul des positions de l image 2 dans l image
fusionne resultante
pos_im2_t = [B2(1,2) - B(1,2) + 1, B2(1,1) - B(1,1) +
1];
pos_im2_b = [ pos_im2_t(1,1) + h2 - 1, pos_im2_t(1,2) +
w2 - 1];

%% On place l image 1 dans l image fusionnee
resultante, en multipliant par la valeur du masque 1
pour que les pixels comptent avec leur "poids fort"
Im(pos_im1_t(1,1):pos_im1_b(1,1),pos_im1_t(1,2):pos_im1
_b(1,2),1:s1) = Im1(1:h1,1:w1,1:s1).*M1(:,:,1);
%% On place l image 2 dans l image fusionnee
resultante, en multipliant par la valeur du masque 2
pour que les pixels comptent avec leur "poids fort"
Im(pos_im2_t(1,1):pos_im2_b(1,1),pos_im2_t(1,2):pos_im2
_b(1,2),1:s2) =
Im(pos_im2_t(1,1):pos_im2_b(1,1),pos_im2_t(1,2):pos_im2
_b(1,2),1:s2) + Im2(1:h2,1:w2,1:s2).* M2(:,:,1);

%% Realisatio du masque resultant avec masque 1
M(pos_im1_t(1,1):pos_im1_b(1,1),pos_im1_t(1,2):pos_im1_
b(1,2)) = M1(1:h1,1:w1);
%% Realisatio du masque resultant avec masque 2
M(pos_im2_t(1,1):pos_im2_b(1,1),pos_im2_t(1,2):pos_im2_
b(1,2)) =
M(pos_im2_t(1,1):pos_im2_b(1,1),pos_im2_t(1,2):pos_im2_
b(1,2)) + M2(1:h2,1:w2);

%% On divise par le masque resultant pour corriger
Im(:,:,:) = Im(:,:,:)./M(:,:,1);
Im(isnan(Im))=0;

end
```

# ANNEXE 3

## *correl_phase .m*

```
%% Projet TS 225 --ì Recalage par correlation de phase |
Application a la construction d'une mosaïque d'images

clear;
close all;
clc;

%% Nous chargeons l image de notre choix
A = double(imread('/phare.jpg'));

w = 256;
h = 128;
tx = 52;
ty = 26;

%% On decoupe l image en 4 sous images
[img1, img2, img4, img6] = decoupe_img(A,w,h,tx,ty);

figure,
subplot(4,1,1);
imshow(uint8(img1));
subplot(4,1,2);
imshow(uint8(img2));
subplot(4,1,3)
imshow(uint8(img4));
subplot(4,1,4)
imshow(uint8(img6));

%% calcul du vecteur translation entre les 4 sous images

[posx, posy] = vect_trans(img1,img2,w,h);
[posx_2, posy_2] = vect_trans(img2,img4,w,h);
[posx_3, posy_3] = vect_trans(img4,img6,w,h);

%% Realisation de la fusion entre les images

[M1, Boite1] = boite_masques(img1);

[M2, Boite2] = boite_masques(img2);
[Boite2] = trans_boite(Boite1, posx, posy);

%% Fusion image 1 / image 2
[img3, M3, Boite3] = fusion( img1, img2, M1, M2, Boite1,
Boite2);

[M4, Boite4] = boite_masques(img4);
[Boite4] = trans_boite(Boite2, posx_2, posy_2);

%% Fusion image 1-2 / image 3
[img5, M5, Boite5] = fusion(img3, img4, M3, M4, Boite3,
Boite4);

[M6, Boite6] = boite_masques(img6);
[Boite6] = trans_boite(Boite4, posx_3, posy_3);

%% Fusion image 1-2-3 / image 4
[img7, M7, Boite7] = fusion(img5, img6, M5, M6, Boite5,
Boite6);

figure,
imshow(uint8(img7));
```

## *decoupe_img.m*

```
function [ img1, img2, img3, img4 ] = decoupe_img(
A,w,h,tx,ty )
%This function cut an image into 2 sub-images with size
w*h and shifted by
%tx and ty
figure,
imshow(uint8(A));

[x, y] = ginput(1);
x = floor(x);
y = floor(y);

%% On effectue le decoupage manuel, en appliquant
toujours tx et ty par rapport ‡ la photo precedente
img1 = A(y:y+h-1,x:x+w-1,:);
img2 = A(y+ty:y+ty+h-1,x+tx:x+tx+w-1,:);
img3 = A(y+2*ty:y+2*ty+h-1,x+2*tx:x+2*tx+w-1,:);
img4 = A(y+3*ty:y+3*ty+h-1,x+3*tx:x+3*tx+w-1,:);

end
```