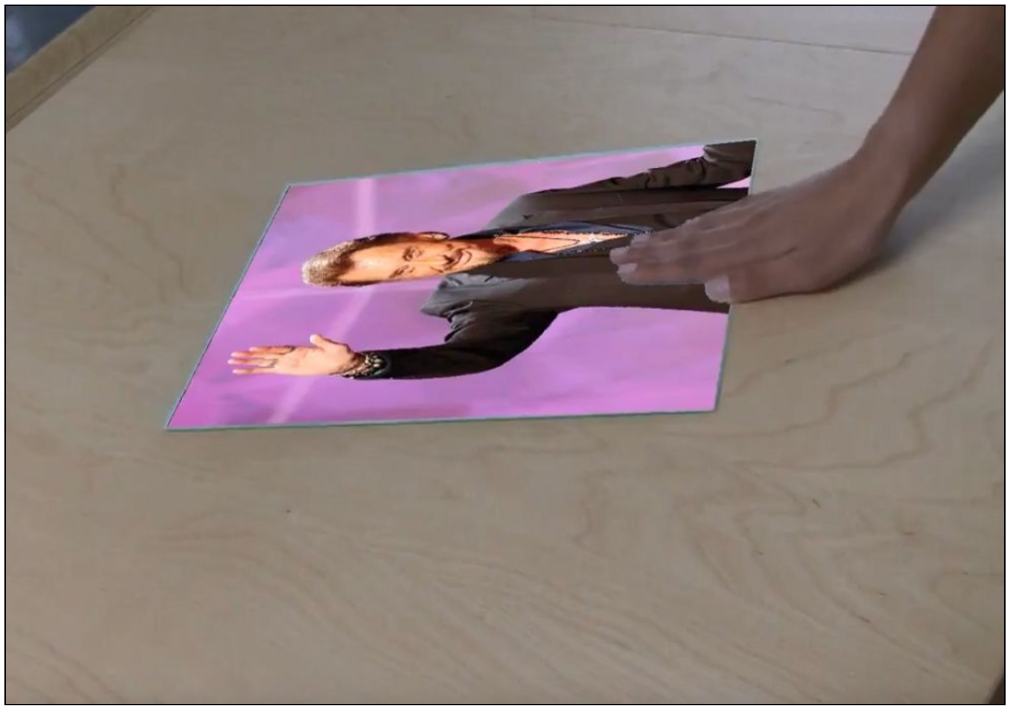
TRAITEMENT DES IMAGES NUMÉRIQUES
Projet de réalité augmentée

- 03/12/2018 -

Partie 1 : Détection du polygone de la feuille



ENSC 2A
COUTHOUIS Fabien
SIOUNET Marie

Projet encadré par M. Marc Donias

Sommaire

I. Introduction

Dans le cadre du cours de traitement des images dispensé en 2nde année de l'ENSC, nous avons réalisé un projet de réalité augmentée, par groupe de quatre.

L'objectif de ce projet est d'effectuer un remplacement de contenu dans une zone précise d'une courte vidéo, à l'aide de Matlab et de nos connaissances acquises durant le cours de traitement des images. Afin de mener ce projet de manière optimale, nous l'avons scindé en 2 parties :

1) **Détection du polygone**
2) **Remplacement de contenu du polygone**

Chaque partie a été traitée par un binôme, puis les travaux ont été mis en communs afin de parvenir au résultat final. Dans ce rapport, nous traiterons uniquement la partie *1) Détection du polygone*, que nous avons effectuée. La partie 2 a été effectuée par le binôme constitué de Victor Garruchet et Hugo Le Tarnec.

Problématique

La vidéo sur laquelle nous devions travailler représentait une feuille rectangulaire de couleur bleue posée sur une table beige. Au cours de la vidéo, une main se positionne sur le bord droit de la feuille et déplace lentement celle-ci sur la table en la faisant glisser.
Notre objectif était de détecter les 4 coins de la feuille tout au long de la vidéo tout en s'accommodant des difficultés techniques telles que :

- *la prise de vue* qui a été réalisée suivant un angle laissant apparaître la feuille non pas comme rectangulaire mais plutôt comme un parallélogramme.
- *la main* qui vient se positionner sur le bord de la feuille et qui ne doit pas être détectée comme un coin, ni disparaître lors du remplacement de contenu.
- *les imperfections du support*, notamment les rainures de la table qui peuvent facilement passer pour des contours.

II. Algorithmique : méthode employée

Nous expliquerons, dans cette partie, la méthode utilisée pour parvenir à détecter les 4 coins de la feuille sur une image de la vidéo originale, puis retrouver ces points à l'image suivante. Cette méthode a été appliquée à toutes les images de la vidéo originale. Notre méthode se découpe en plusieurs parties.

1. Détection de contour : calcul du gradient d'intensité

Nous nous sommes basés sur l'approche de Canny pour la détection de contour. Nous avons ainsi calculé les gradients d'intensité de l'image (dans les directions horizontales et verticales), par convolution avec la dérivée d'une Gaussienne.

En notant I l'image de départ, les dérivées horizontales (I_x) et verticales (I_y) s'obtiennent alors comme noté ci-dessous :

$$I_x = I * G^x_{\sigma_{dx}\sigma_{dy}} = I * \frac{-x}{2\pi\sigma^3_{dx}\sigma_{dy}} exp(-(\frac{x^2}{2\sigma^2_{dx}} + \frac{y}{2\sigma^2_{dy}}))$$

$$I_y = I * G^y_{\sigma_{dx}\sigma_{dy}} = I * \frac{-y}{2\pi\sigma^3_{dy}\sigma_{dx}} exp(-(\frac{x^2}{2\sigma^2_{dx}} + \frac{y}{2\sigma^2_{dy}}))$$

Par la suite, pour simplifier, nous prendrons : $\sigma_d = \sigma_{dx} = \sigma_{dy}$ (variance de la Gaussienne utilisée pour la calcul des dérivées directionnelles).

Taille du filtre

La taille du filtre gaussien est choisie grâce à la *règle des* 3σ, qui stipule que presque toutes les valeurs d'une loi normale se situent dans un intervalle centré autour de la moyenne et dont les bornes se situent à 3 écarts-types de part et d'autre.

On a donc un filtre de la forme : $[-3\sigma, 3\sigma]$.

Soit un filtre de taille $6\sigma + 1$.

2. Détecteur de Harris

a) Généralités

Le détecteur de Harris sert à détecter les zones d'intérêt sur une image. Nous passerons sous silence les détails mathématiques pour nous concentrer sur l'aspect algorithmique.

On notera :

$$I_x^2 = I_x \times I_x$$

$$I_y^2 = I_y \times I_y$$

$$I_{xy} = I_x \times I_y = I_{yx}$$

Considérons C, la matrice d'autocorrélation suivante :

$$C = \begin{pmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{pmatrix}$$

On note R la réponse du détecteur de Harris, fonction définie par :

$$R_\sigma(x,y) = det(C(x,y)) - k \times Tr(C(x,y))^2$$

Avec : k une constante empirique appartenant à $[0.04, 0.15]$

On peut ainsi calculer la valeur de R en chaque point de l'image.

Les valeurs de R sont :

- Positives au voisinage d'un coin
- Négatives au voisinage d'un contour
- Faibles ou nulles dans une région d'intensité constante

On en déduit donc que les coins sont représentés par les maximas locaux de R.

b) Notre approche

Nous avons d'abord appliqué un filtre gaussien sur les composantes de la matrice C, ceci afin de réduire au maximum le bruit et les faux positifs. La taille du filtre est choisie selon la règle des 3σ, expliquée précédemment.

On note la fonction gaussienne comme suit :

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \times exp(\frac{x^2+y^2}{2\sigma^2})$$

Ainsi, nous pouvons calculer les composantes "lissées" de la matrice C ("lissées" = convoluées par notre fonction gaussienne).

$$S_x^2 = I_x^2 * G_\sigma$$

$$S_y^2 = I_y^2 * G_\sigma$$

$$S_{xy} = I_{xy} * G_\sigma$$

Nous obtenons ainsi la matrice d'autocorrélation suivante :

$$C' = C_{lissée} = \begin{pmatrix} S_x^2 & S_{xy} \\ S_{xy} & S_y^2 \end{pmatrix}$$

On peut ensuite calculer :

$$R'(x,y) = det(C'(x,y)) - k \times Tr(C'(x,y))^2$$

Précisons que les valeurs des variances utilisées ont leur importance dans la détection, puisqu'elles définissent la taille et les valeurs du filtre gaussien utilisé pour le calcul des dérivées, ainsi que celui utilisé pour le lissage des dérivées.

Dans ce qui suit, pour simplifier, on notera la matrice H, définie en chaque point de l'image initiale I, comme ci-dessous :

$$H(I,\sigma_g,\sigma_d,k) = det(C') - k \times Tr(C')^2$$

Avec :

- I : image initiale

- σ_g : variance utilisée pour le lissage gaussien
- σ_d : variance utilisée pour calculer les dérivées directionnelles de l'image
- k : constante empirique appartenant à $[0.04, 0.06]$

3. Détection des coins de la feuille : détecteur utilisé

Comme dit précédemment :

Les valeurs de H sont :
- Positives au voisinage d'un coin
- Négatives au voisinage d'un contour
- Faibles ou nulles dans une région d'intensité constante

Chaque coin sera donc représenté par un maximum local de H.

Afin de réduire le bruit au maximum, notre détecteur sera formé par un produit (termes à termes) de 2 détecteurs de Harris, avec des valeurs de variance différentes.

Notons également qu'un produit de deux nombres négatifs est positif. Or, nous recherchons les coins (et non les contours). Nous devons donc prendre la valeur absolue des termes d'un des deux détecteurs pour éviter de détecter des contours.

Nous pouvons donc construire notre détecteur D comme suit:

$$D = H(I, \sigma_g, \sigma_{d1}, k) \times \left| H(I, \sigma_g, \sigma_{d2}, k) \right|$$

Avec :
- I : image initiale
- $\sigma_g = \sigma_{g1} = \sigma_{g2}$: même variance de lissage gaussien pour les 2 détecteurs
- σ_{d1} : utilisée pour le calcul des dérivées directionnelles de l'image pour le détecteur 1
- σ_{d2} : utilisée pour le calcul des dérivées directionnelles de l'image pour le détecteur 2
- k : constante empirique appartenant à $[0.04, 0.15]$

Les maximas locaux de cette matrice représentent les coins de l'image.
Les 4 coins de la feuille sont donc normalement représentés par les maximas locaux d'une fenêtre entourant ces coins. Cependant, il est possible que cela ne soit pas toujours le cas. C'est pourquoi, lors du passage à l'image suivante de la vidéo, chaque coin de la feuille doit être recalculé intelligemment, de sorte à ne pas détecter autre chose (main, bruit, tâche sur la table, …).

4. Détection des coins de la feuille : prédiction et calcul des maximas

Le problème qui se pose dans cette partie est le suivant :
Comment, à partir des positions des 4 coins de la feuille à un instant t, trouver les positions de ces coins à un instant t+1 ?

La solution théorique choisie est la suivante :
Connaissant la position d'un coin de la feuille à un instant t et à un instant t-1, on suppose que le coin suivra plus ou moins la même trajectoire à un instant t+1. On définit ainsi la position prédite du point à un instant t+1 :

$$x_{t+1}^{prédit} = x_t + \frac{x_t - x_{t-1}}{2}$$

x_{t+1} sera normalement le maximum local de H dans une fenêtre autour de ce point $x_{t+1}^{prédit}$.

La difficulté ici sera ici de choisir la taille de la fenêtre : assez grande pour détecter le point x_{t+1} mais pas trop grande pour ne pas détecter un autre coin (main, défaut sur la table, etc.).

III. Réalisation

Cette partie traitera de la réalisation de ce projet sous Matlab. Les conventions de nommage utilisées seront identiques à celles utilisées dans la partie précédente.

1) Organisation du code

Nous avons choisi de décomposer notre code en plusieurs fonctions :

- **Gradient** : Calcul des dérivées directionnelles de l'image de départ en fonction d'une variance σ_d donnée.
- **Gaussian** : Calcul de la gaussienne correspondant à une variance σ_g donnée.
- **Harris** : Calcul de l'image passée au détecteur de Harris. Cette fonction fait appel à la fonction *gaussian* pour le lissage des dérivées, et nécessite de passer en paramètre

les dérivées directionnelles de l'image, ainsi que la variance σ_g utilisée pour l'appel de Gaussian.

- **Detector** : Détecteur utilisé pour notre projet, composé des 2 Harris utilisant des variances différentes. Cette fonction fait appel à la fonction *gradient* et à la fonction *harris*
- **TransformationVideo :** Fonction fournie en partie par l'autre binôme du groupe, permettant de mettre en commun notre travail pour générer la vidéo finale.
- **Exécutable :** Script commun à exécuter pour la génération de la vidéo finale, avec remplacement de l'image.

2) Valeurs des divers paramètres

➢ Variance

Les valeurs de variances utilisées sont les suivantes (conventions de nommage identiques à la partie II de ce rapport : **II Algorithmique : méthode employée**).

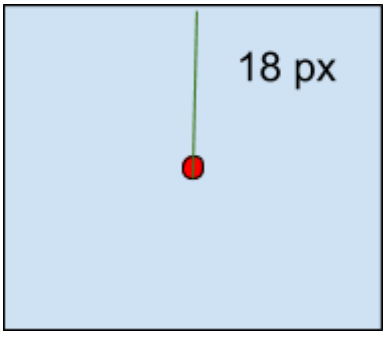
Détecteur de Harris 1 : $\sigma_d = 2;\ \sigma_g = 3\ ;\ k = 0,05$

Détecteur de Harris 2 : $\sigma_d = 2;\ \sigma_g = 5\ ;\ k = 0,05$

Ces valeurs, proches de celles recommandées par le professeur ont permis d'éviter la détection de faux positifs, tout en conservant une bonne détection des coins de la feuille.

➢ Taille de la fenêtre

Là encore, la valeur proposée par le professeur s'est révélée être satisfaisante. Nous avons donc choisi une taille de fenêtre correspondant à une zone de 18 pixels autour du point prédit.



3) Points particuliers

Cette partie liste tous les points particuliers/intéressants que comporte notre implémentation.

- ✓ L'image est convertie en *double* et représentée en niveaux de gris au début de la fonction Detector. Cela permet (en plus de gagner en performance), de pouvoir utiliser la fonction *conv2* de Matlab (produit de convolution.

```
function [D] = detector(I)
I = double (I);
I = (I(:,:,1)+I(:,:,2)+I(:,:,3))/3;
```

- ✓ La taille des filtres gaussien (ainsi que sa dérivée pour le calcul des gradients) ont été arrondi à la valeur supérieur (en cas d'utilisation de variance non entière).

```
function [G] = gaussian(sigmag)
    filterSize = ceil(3*sigmag);
   [X,Y]=meshgrid(-filterSize:filterSize);
```

- ✓ Les appels à la fonction *conv2* de Matlab ont été faits en utilisant l'attribut 'same' :

```
Gx=conv2(I,Hx,'same');
```

  Ceci permet d'obtenir en sortie une matrice de la même taille que la première matrice passée en paramètre (I ici). Sans ce 'same', la matrice obtenue en sortie de convolution serait plus grande, ce qui complexifierait inutilement les calculs.

- ✓ Les indices de matrice correspondant aux pixels de l'image ont été, au besoin, arrondi à la valeur inférieure.

- ✓ Les calculs utilisant des boucles ont été limités au maximum afin de garantir une vitesse d'exécution satisfaisante sous Matlab.

Le code commenté vous est présenté en annexe pour plus d'informations.

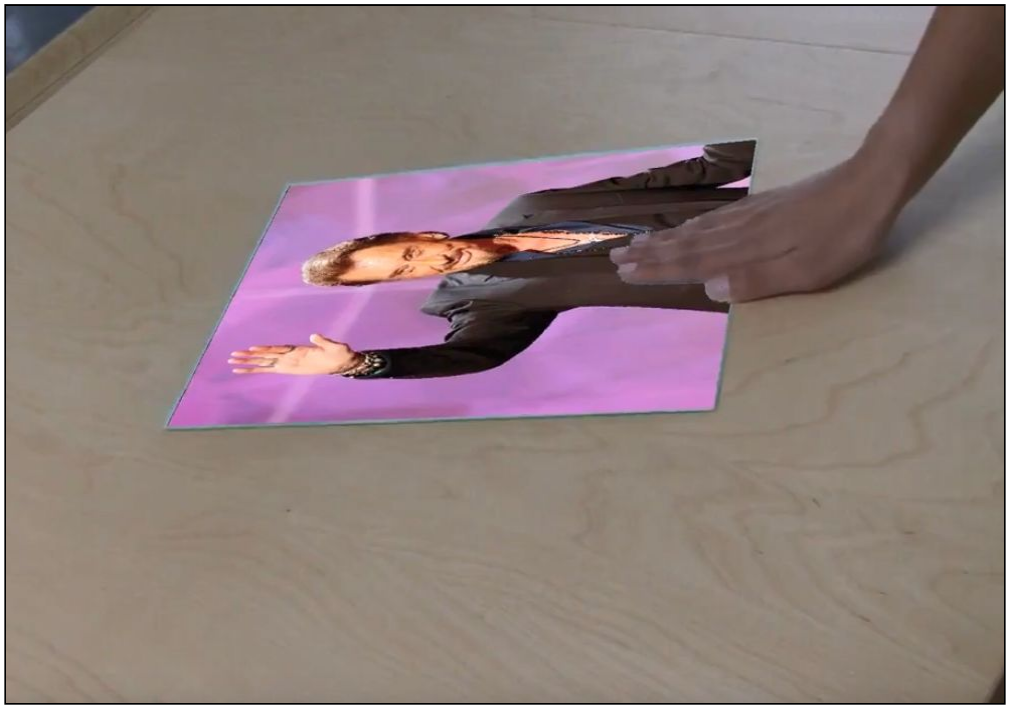
IV. Résultats

1) Résultats du binôme

L'objectif de notre binôme était de détecter les coins de la feuille tout au long de la vidéo de la manière la plus précise possible. Nous nous sommes d'abord intéressés à la détection de coins uniquement sur la première image puis nous nous sommes attelés à la détection sur l'ensemble de la vidéo. Cette étape a été fastidieuse car nous observions parfois des changements importants entre 2 images successives, ce qui rendait rapidement nos résultats inexploitables.
Nous sommes cependant parvenus à résoudre ce problème, ce qui a permis à Victor Garruchet et Hugo Le Tarnec de finaliser le remplacement de contenu.

2) Résultats du quatuor

L'ambition du groupe était de remplacer le contenu de la feuille par une image de notre choix et si possible d'ajouter un contenu 3D en nous appuyant sur les objets disposés sur la feuille.
Nous ne sommes pas parvenus à réaliser la seconde partie (optionnelle) mais avons obtenu des résultats concluants concernant le remplacement du contenu tout en préservant le positionnement de la main comme on peut l'observer sur l'image ci-dessous.



V. Conclusion

Malgré quelques appréhensions, nous avons pas à pas réussi à cerner le projet dans sa globalité et à nous familiariser avec l'environnement de travail Matlab. Nous avons ainsi fini par obtenir le résultat escompté. Néanmoins, il reste des points sur lesquels une optimisation serait encore possible, notamment en ce qui concerne le temps d'exécution ; bien que celui-ci soit satisfaisant (étant donné la basse qualité ainsi que la courte durée de la vidéo), il aurait pu être intéressant d'envisager le calcul du détecteur de Harris uniquement sur la fenêtre autour du point prédit afin de réduire davantage cette durée d'exécution.

Finalement, à défaut d'être parvenus à réaliser l'ensemble des exigences attendues, notamment concernant le contenu 3D, nous espérons tout de même avoir rendu un bel hommage à celui qui nous donne chaque jour l'envie d'avoir envie.

VI. Annexes : Code

Executable.m

```
clear all;
close all;
clc;

v = VideoReader('video.mp4');
nbreTrame = v.Duration*v.FrameRate;
[imIncruste] = imread('jo.jpg');

video = transformationVideo(v, nbreTrame, imIncruste);

```

transformationVideo.m

```
function[mov]=transformationVideo(video, nbreTrame, imIncruste)
    %Initialisation
    mov = VideoWriter('F_1.avi');
    open(mov);
    [imBase] = read(video,1);

    %Positionnement des coins initiaux de la feuille
    figure; imshow(imBase);
    nbPoints = 4;
    [XNow,YNow]=ginput(nbPoints);
    %Initialisation des variables à l'état initial (image 1 de la vidéo)
    XNext=XNow ; YNext=YNow;

    for i = 1:nbreTrame
       % Définition des images
        [imBase] = read(video,i);

        % Définition des points des 2 images
        XPrev = XNow;
        YPrev = YNow;
        XNow = XNext;
        YNow = YNext;
        D=detector(imBase);
        [XNext,YNext] = nearestCorner(D,nbPoints,XNow,YNow,XPrev,YPrev);
        [XNext,YNext] = nearestCorner(D,nbPoints,XNow,YNow,XPrev,YPrev);

        % On récupère la main
        imHand = PutHand(imBase,imIncruste,YNext,XNext);

        xInc1 = 0;
        yInc1 = 0;
        xInc2 = 0;
        yInc2 = size(imHand,2);
        xInc3 = size(imHand,1);
        yInc3 = 0;
        xInc4 = size(imHand,1);
        yInc4 = size(imHand,2);

        xIncVect = [xInc1,xInc2,xInc3,xInc4];
        yIncVect = [yInc1,yInc2,yInc3,yInc4];

        %Matrice d'homographie
        H = homographyEstimate(YNext,XNext,xIncVect,yIncVect);

        imFinale = homographyProjection(H,imBase,imHand); % Incrustation

        writeVideo(mov, imFinale); % On met l'image dans la video finale
    end
    close(mov); %Fermeture du fichier
end
```

detector.m

```
%Detecteur de coins utilisant un double détecteur de Harris
    function [D] = detector(I)
    I = double (I);
    I = (I(:,:,1)+I(:,:,2)+I(:,:,3))/3;

    sigmad=2;
    sigmag1 = 3;
    sigmag2 = 5;

    [Ix,Iy] = gradient(I,sigmad);
    H1 = harris(Ix,Iy,sigmag1);
    H2 = harris(Ix,Iy,sigmag2);

    D = H1.* abs(H2);
end
```

gradient.m

```
function [Gx,Gy] = gradient(I, sigmad)
    filterSize = ceil(sigmad*3);
    [X,Y]=meshgrid(-filterSize:filterSize);
    Hx = -X.* exp( -(X.^2 + Y.^2)/(2*sigmad^2) ) / (2*pi*sigmad^4) ;
    Hy = -Y .* exp( -(X.^2 + Y.^2)/(2*sigmad^2) ) / (2*pi*sigmad^4);

    Gx=conv2(I,Hx,'same');
    Gy=conv2(I,Hy,'same');
end
```

harris.m

```
function [H] = harris(Ix,Iy, sigmag)
    k = 0.05; % constante appartenant à [0.04;0.06] (empiriquement)

    % Lissage des images Ix2, Iy2, Ixy (application filtre gaussien)
    Gauss = gaussian(sigmag);

    Ix2 = Ix.^2;
    Iy2 = Iy.^2;
    Ixy = Ix.*Iy;

    Sxx=conv2(Ix2,Gauss,'same');
    Syy=conv2(Iy2,Gauss,'same');
    Sxy=conv2(Ixy,Gauss, 'same');

%     % Opérateur de Harris (det(C) - tr(C)^2)
     H = Sxx.*Syy-Sxy.^2 - k*(Sxx+Syy).^2;
end
```

gaussian.m

```
function [G] = gaussian(sigmag)
    filterSize = ceil(3*sigmag);
    [X,Y]=meshgrid(-filterSize:filterSize);
    G = exp(-(X.^2+Y.^2)/(2*sigmag^2))/(2*pi*sigmag^2);
end
```

nearestCorner.m

```
function [XNext, YNext] = nearestCorner(I,nbPoints, XNow,YNow,XPrev,YPrev)
%Renvoie l'emplacement du pixel à l'étape n+1, connaissant sa position
%aux étapes n (x2,y2)  et n-1 (x1,y1)

    %Demi taille de la fenêtre de recherche
    windowSize = 18;
    XPred = zeros(1,nbPoints);
    YPred = zeros(1,nbPoints);
    XNext = zeros(1,nbPoints);
    YNext = zeros(1,nbPoints);

    %fenêtre de I étudiée
    for p=1:nbPoints
        %Point prédit
        XPred(p) = floor(XNow(p) + (XNow(p)-XPrev(p))/2);
        YPred(p) = floor(YNow(p) + (YNow(p)-YPrev(p))/2);

        studiedWindow = I(YPred(p)-windowSize:YPred(p)+windowSize,XPred(p)-windowSize:XPred(p)+windowSize);
         maxInRadius = max(max(studiedWindow));
        [YNext(p),XNext(p)] = find(studiedWindow == maxInRadius);

         %On attend un pixel entiers en sorties
        %On pense à renvoyer les pixels de I et non de la fenêtre de I étudiée
        XNext(p) = floor(XNext(p)+XPred(p)-windowSize);
        YNext(p) = floor(YNext(p)+YPred(p)-windowSize);
    end
end
```