République Algérienne Démocratique et Populaire

Ministaire de l'Enseignement Supérieur et de la Recherche Scientifique

École Nationale Polytechnique

Département d'Électronique

Projet de fin d'etude en vue de l'obtention du Diplome de Master en Électronique

Étude théorique et implémentation de l'algorithme SIFT

Réalisé par :
OUDNI Louiza

Proposé et Encadré par :
Pr C.Larbes
Pr A.Bouridane

Promotion 2013

ملخص:

الهدف من هذا المشروع هو دراسة خوارزميةSIFT لمعالجة الصور.

الخوارزمية المستعملة في هذا المشروع صممت خصيصا لتتبع الأشياء.

زيادة عن دراسة لمختلف الوسائل المستعملة و التطبيقات SIFTقمنا بتطبيق استقطاب نقاط مهمة من الصور بواسطة ماتلابMATLAB .

الكلمات المفتاحيّة:رؤية بواسطة الحاسب، الكشف، واصف ، نقطة التوقف ،تتبع ، فيديو.

Résumé

Dans ce projet, nous avions pour but l'étude de l'algorithme de traitement d'image, le SIFT.

Il s'agit d'un algorithme réalisé spécialement pour la reconnaissance d'objet sur des images et la mise en correspondance de celle-ci.

En plus de l'étude des différents outils utilisés et des applications du SIFT, une implémentation de la partie extraction des points d'intérêt à l'aide de Matlab a été réalisée.

Mots clés : Vision par ordinateur, SIFT, Reconnaissance, mise en correspondance, Point d'intérêt, Descripteurs.

Abstract

In this project, we aim to study the image processing algorithm, SIFT.

It is an algorithm designed specifically for object recognition and matching between images.

In addition to the study of various SIFT tools and applications, an implementation of the keypoints extraction was conducted in using Matlab.

Keywords : Computer Vision, SIFT, recognition, matching, point of interest descriptors.

Remerciements

Je tiens d'abord à exprimer ma gratitude à mes promoteurs, au Pr.BOURIDANE pour m'avoir initiée au domaine du traitement d'image, et au Pr.LARBES pour la disponibilité et l'écoute permanentes dont il a fait preuve tout au long de ce projet.

Je tiens à remercier les membres du jury pour l'intérêt accordé à mon travail.

Enfin, Je remercie tous ceux qui auront contribué à ce travail et à mon cursus universitaire.

Dédicaces

Je dédicace ce travail à :
Ma très chère famille,
Mes chers amis,
Aux membres de la famille de l'Ecole Polytechnique.

Louiza

Table des matières

Table des figures

Liste des tableaux

Abbreviations

DoG	**D**ifference **o**f **G**aussians
SIFT	**S**cale **I**nvarant **F**eature **T**ransform

Introduction générale

En intelligence artificielle, la vision par ordinateur est au cœur de nombreuses applications. Donnons quelques exemples. La vision par ordinateur peut être un outil pour la reconnaissance d'objet, le suivi ou le tracking, la reconstruction 3D.

L'objectif de ce projet est l'étude et l'implémentation d'un algorithme de traitement d'image, dont la fonction principale est la reconnaissance et l'appariement d'images. Il s'agit de l'algorithme SIFT.

Dans le premier chapitre, nous nous intéressons à l'étude théorique de cet algorithme, aux outils utilisés par celui-ci, et aux applications pour lesquelles il a été conçu.

Dans le chapitre suivant, les détails de l'implémentation du SIFT sur Matlab sont expliqués. Il y a aussi une les résultats obtenus grâce à cette implémentation.

Chapitre 1

SIFT : L'algorithme

1.1 Introduction

Le SIFT est un algorithme robuste vis-à-vis des changements de l'image. Il est en l'occurrence invariant par rapport aux changements d'échelle, de la luminosité. Les points clés détectés d'une image prise sous des points de vue différents, restent stables. De ce fait, les applications de cet algorithme sont nombreuses.

1.2 Les applications du SIFT

1.2.1 Appariement des images

Le SIFT permet de dire si deux images représentent le même objet ou représentent des parties d'une même scène. L'image suivante illustre cela.

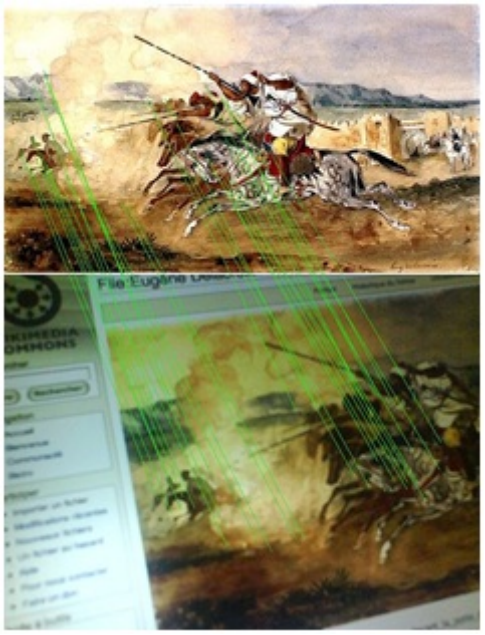

FIGURE 1.1: Mise en correspondance de deux images avec SIFT

Nous pouvons remarquer sur l'image ci-dessus, que malgré les points de vue différents des prises des deux images, le SIFT parvient à relier les points communs.

1.2.2 Assemblage des panoramas

Par exemple, il est très utilisé pour l'assemblage des panoramas. C'est-à-dire, qu'à partir de plusieurs images représentant des parties d'une même scène, il peut reconstruire une image panoramique.

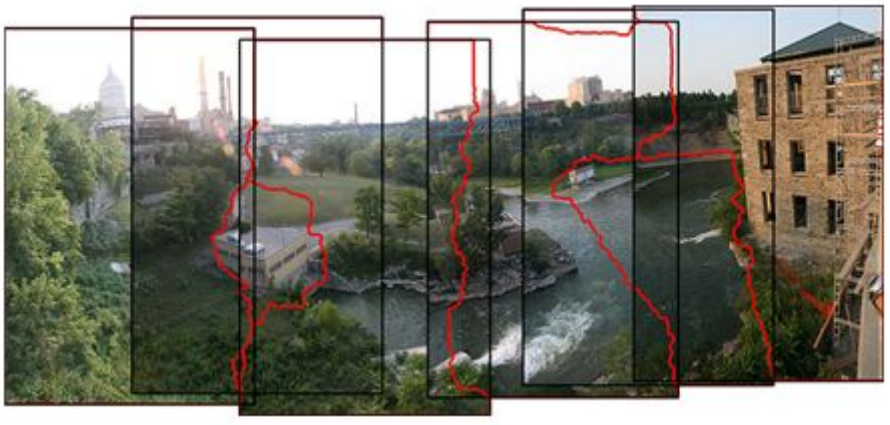

FIGURE 1.2: Assemblage d'un panorama avec SIFT

1.2.3 Réalité augmentée

La réalité augmentée est une nouvelle tendance de la vision par ordinateur. Il s'agit d'ajouter des objets virtuels à une scène réelle pour construire une scène mixte. Cela se fait en temps réel. La précision des caractéristiques SIFT extraites de l'image, permet d'obtenir des résultats satisfaisants dans ce domaine d'application.

1.3 Les étapes de l'algorithme

1.3.1 Détection des points d'intérêts

1.3.1.1 Construction de l'espace-échelle gaussien

Les points d'intérêts SIFT correspondent aux extrema locaux des différences de filtres gaussiens à différentes échelles.

La construction de cet espace de différences de Gaussiennes se fait en deux temps. Tout d'abord l'image en niveaux de gris est convoluée à plusieurs filtres gaussiens, comme suit :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1.1}$$

Avec
$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$
$I(x, y)$: L'image en niveaux de gris.
x, y : Coordonnées des pixels de l'image.
σ : Ecart-type du filtre Gaussien.
Les valeurs des coefficients du noyau prennent donc des valeurs proportionnelles à $G(x, y, \sigma)$ [1].
Par exemple,pour un filtre de $\sigma = 1.6$, et de taille de noyau 7, les coefficients du noyaux sont :

0.0052	0.0137	0.0247	0.0300	0.0247	0.0137	0.0052
0.0093	0.0247	0.0444	0.0539	0.0444	0.0247	0.0093
0.0113	0.0300	0.0539	0.0656	0.0539	0.0300	0.0113
0.0093	0.0247	0.0444	0.0539	0.0444	0.0247	0.0093
0.0052	0.0137	0.0247	0.0300	0.0247	0.0137	0.0052
0.0019	0.0052	0.0093	0.0113	0.0093	0.0052	0.0019

TABLE 1.1: Coefficients du noyau d'in filtre gaussien de taille 5 avec $\sigma = 1.6$

La figure 1.3 illustre les résultat du filtrage de la célébre image en traitement d'image de Lena par un filtre gaussion de $\sigma = 4$ et taille du masque 25.

FIGURE 1.3: Effet sur l'image pour $\sigma = 4$ et taille du masque 25
a)Image originale b)Image filtrée.

Pour la construction de cet espace-echelle,le facteur d'échelle est fixé à une valeur initiale σ_0 , et augmenté à chaque fois. Plus exactement, l'échelle est multipliée par un facteur k. Ce facteur est déterminé par le nombre s d'images qu'on veut obtenir par octave dans l'espace-échelle gaussien suivant la relation $k = 2^{1/s}$.
Lorsque la valeur $2\sigma_0$ est atteinte :

- Les images filtrées jusqu'à présent constituent une octave.
- Les dimensions de l'image sont réduites de moitiés, et l'algorithme est reproduit de nouveau avec cette image réduite pour obtenir une seconde octave.

- Le calcul des images filtrées est arrêté lorsque les dimensions de l'image deviennent très petites.

A la fin de cette étape nous obtenons un ensemble d'images floues à des échelles différentes, comme illustrées a gauche de la figure 1.4.
Une fois, les octaves obtenues, l'étape suivante consiste à calculer les différences entre deux images successives :

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{1.2}$$

L'ensemble des différences de gaussienne $D(x, y, \sigma)$ constitue l'espace-échelle gaussien. Comme illustré à droite de la figure 1.4.
D. Lowe recommande la valeur $\sigma_0 = 1.6 * k$.[2]
Nous devons produire $S = s + 3$ images dans la pile d'images floues pour chaque octave, de sorte que la détection des extrema, abordée au paragraphe suivant puisse couvrir une octave complète.

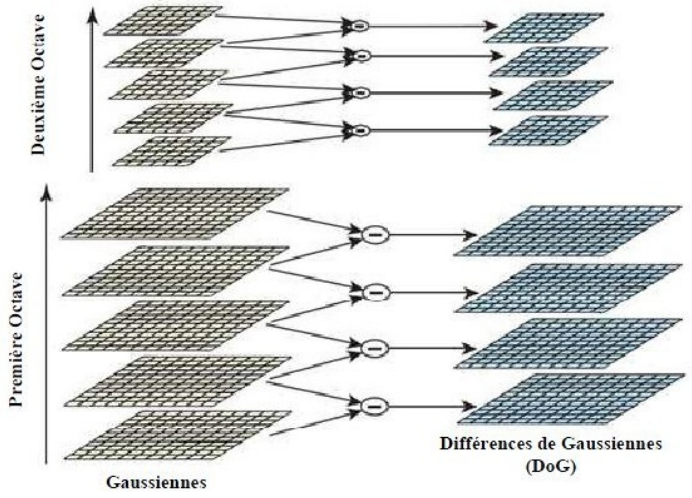

FIGURE 1.4: Illustration de l'espace-échelle Gaussiennes et differences de Gaussiennes

Les figures 1.5 et 1.6 illustrent respectivement l'espace-échelle Gaussiennes et diferrences de Gaussiennes de l'image Lena illustrée au a) de la figure **??**

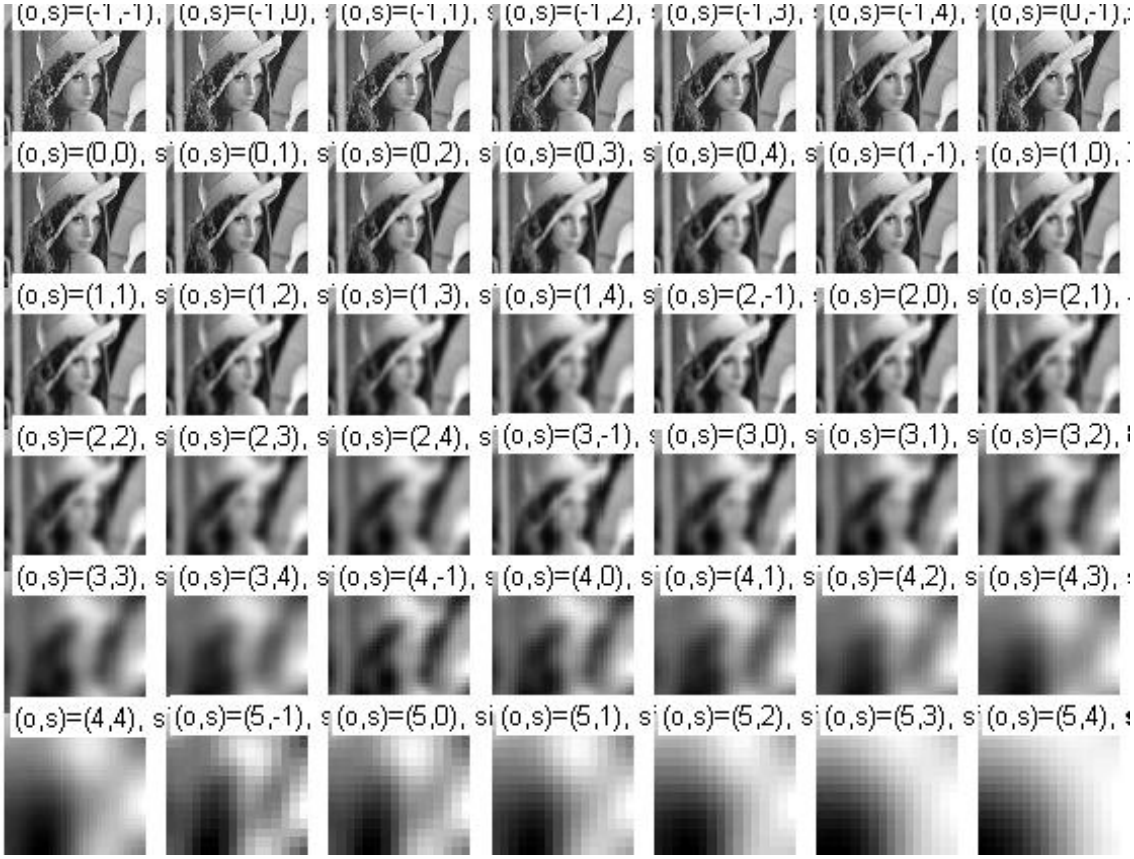

FIGURE 1.5: Illustration

1.3.1.2 Localisation des extrema locaux

Une fois l'espace des différences de Gaussiennes obtenu, il suffit de calculer les extrema locaux. Ces extrema sont sélectionnés de la manière suivante : Chaque pixel est comparé avec ses 26 voisins (8 pixels voisins sur la même image, 9 pixels sur chacune des images au dessus et en dessous de son image dans l'espace-échelle gaussien)comme illustré sur la figure 1.7.
Le pixel est sélectionné seulement si c'est le maximum ou le minimum de tous ses voisins. Ces extrema sont des points clés potentiels. Les étapes suivantes vont permettre de les relocaliser avec précision et d'en éliminer un certain nombre.

1.3.1.3 Amélioration de la précision par interpolation des coordonnées

Lorsqu'un point d'intérêt est localisé sur un étage de l'espace échelle, différent du premier étage, une optimisation sous-pixelique est effectuée, afin de positionner au mieux ce pixel sur l'image de taille initiale.

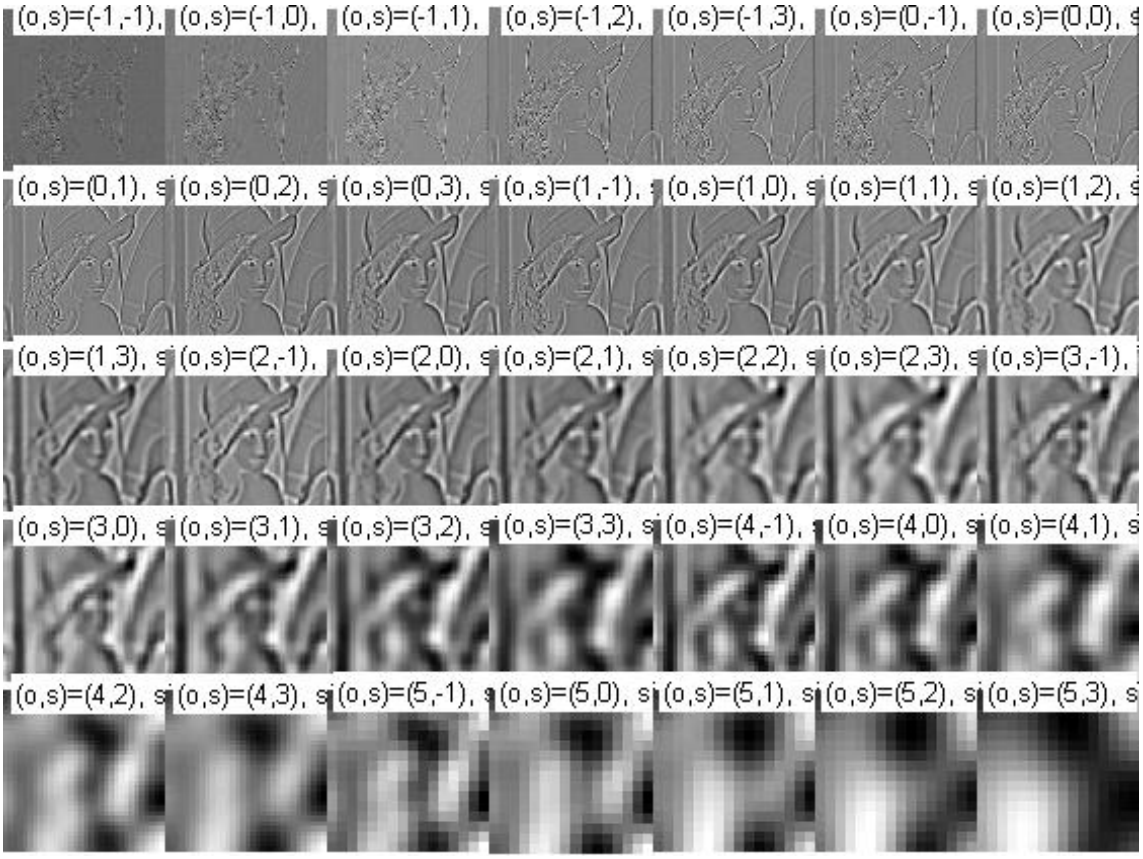

FIGURE 1.6: Illustration

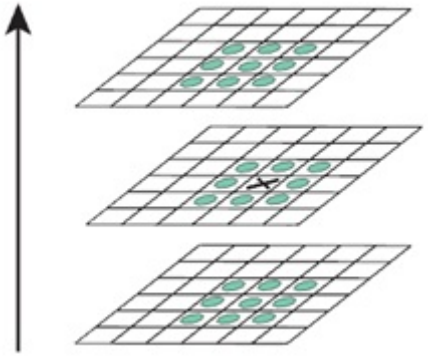

FIGURE 1.7: Localisation des extrema locaux

Cela s'obtient par un développement de Taylor d'ordre 2 de la fonction $D(x, y, \sigma)$, en prenant comme origine les coordonnées du point d'intérêt candidat :

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x^T}\frac{\partial^2 D^T}{\partial \mathbf{x}^2}\mathbf{x} \tag{1.3}$$

où $\mathbf{x} = (x, y, \sigma)^T$au voisinage du point d'intérêt——
La position précise de l'extremum $\hat{\mathbf{x}}$ est déterminée en résolvant l'équation annulant la dérivée de cette fonction par rapport à $\mathbf{x}$:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1}\frac{\partial D}{\partial \mathbf{x}} \tag{1.4}$$

Si le $\hat{\mathbf{x}} > 0.5$ dans l'une des trois dimensions, cela signifie que le point est plus proche d'un des voisins dans l'espace des échelles discret.
Dans ce cas, le point d'intérêt candidat est mis à jour et l'interpolation est réalisée à partir des nouvelles coordonnées. Sinon, le delta est ajouté au point candidat initial qui gagne ainsi en précision.

1.3.1.4 Élimination des points d'intérêts de faible contraste

La valeur de la fonction $D(\hat{\mathbf{x}})$ est utile pour l'élimination d'extrema avec faible contraste. En combinant les deux équations précédentes nous trouvons :

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}} \tag{1.5}$$

Un seuillage absolu ($|D(\hat{\mathbf{x}})| < 0.03$) est effectué pour éliminer les points instables, à faible contraste.

1.3.1.5 Élimination des points situés sur les arêtes

Les points situés sur les arêtes (ou contours) doivent être éliminés. Ceci, car la fonction $D(x, y, \sigma)$ y prend des valeurs élevées, ce qui donne naissance à des extrema locaux instables, très sensibles au bruit.
Un point instable aura une grande courbure le long du contour, mais une faible courbure dans la direction perpendiculaire. Afin d'éliminer ces points, une matrice Hessienne 2x2

est calculée à la position et échelle du point clé :

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{1.6}$$

Numériquement **H**est calculée par les formules suivantes :

$$D_{xx} = 0.5(D(x+1, y, \sigma) + D(x-1, y, \sigma) - 2 * D(x, y, \sigma)) \tag{1.7}$$

$$D_{yy} = 0.5(D(x, y+1, \sigma) + D(x, y-1, \sigma) - 2 * D(x, y, \sigma)) \tag{1.8}$$

$$D_{xy} = 0.25(D(x+1, y+1, \sigma) + D(x-1, y-1, \sigma) - D(x-1, y+1, \sigma) - D(x+1, y-1, \sigma)) \tag{1.9}$$

Les dérivées partielles sont estimées en effectuant les différences des voisions du point clé. Les valeurs propres de la matrice Hessienne sont proportionnelles aux courbures principales de D.
Mais comme nous nous intéressons uniquement au rapport des deux valeurs propres, il est inutile de les calculer, puisque la trace et le déterminant de la matrice permettent de déduire respectivement la somme et le produit de ces deux valeurs.

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta \tag{1.10}$$

$$Det(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \tag{1.11}$$

En supposant que α est la plus grande valeur propre et r le rapport entre la plus grande et la plus petite valeur propre $r = \alpha/\beta$
Nous avons :

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{1.12}$$

Comme la fonction $\frac{(r+1)^2}{r}$ est strictement croissante sur $[1, +\infty]$, donc pour vérifier que le rapport r est au dessus d'un certain seuil r_{seuil}, il suffit de vérifier que :

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r_{seuil}+1)^2}{r_{seuil}} \tag{1.13}$$

Ceci est beaucoup moins coûteux en nombre d'opérations, que le calcul des valeurs propres.
Lowe recommande de fixer r_{seuil} à 10 , et donc d'éliminer les points clés où le rapport des deux principales courbures est supérieur à 10 [2].

1.3.2 Calcul des descripteurs

1.3.2.1 Assignation d'orientation

Maintenant que les points clés sont déterminés, la présente étape est la dernière avant le calcul des descripteurs. Elle permet d'attribuer à chacun une ou plusieurs orientations déterminées localement sur l'image. C'est ce qui assurera l'invariance de la méthode part rapport à la rotation et au changement d'échelle.

Pour un point clé donné (x_0, y_0, σ_0), nous calculons d'abord la norme $m(x, y)$ et l'orientation $\theta(x, y)$. Le calcul s'effectue au niveau de ses points voisins sur l'image filtrée, calculée au départ $L(x, y, \sigma)$ avec σ_0 le plus proche du facteur d'échelle du point :

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{1.14}$$

$$\theta(x,y) = tan^{-1}\frac{(L(x+1,y) - L(x-1,y))}{(L(x,y+1) - L(x,y-1))} \tag{1.15}$$

Une fois ce calcul préliminaire effectué, un histogramme des orientations est réalisé avec des intervalles couvrant chacun 10 degrés d'angle(figure 1.8). L'histogramme est doublement pondéré : d'une part, par une fenêtre circulaire gaussienne de paramètre égal à $1.5\sigma_0$, d'autre part, par l'amplitude de chaque point.
Les pics dans cet histogramme correspondent aux orientations dominantes. Toutes les orientations dominantes permettant d'atteindre au moins 80% de la valeur maximale sont prises en considération. Ce qui provoque si nécessaire la création de points-clés supplémentaires ne différant que par leur orientation principale.
À l'issue de cette étape, un point-clé est donc défini par quatre paramètres(x, y, σ, θ).

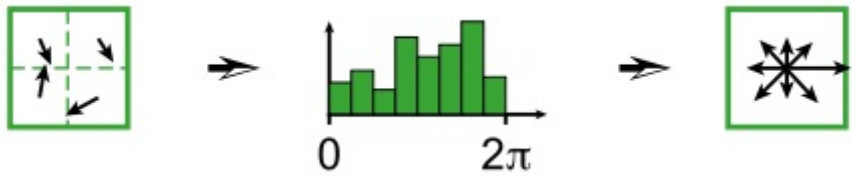

FIGURE 1.8: Construction de l'histogramme des orientations d'un point clé

La figure 1.9 illustre quelques points clé de l'image Lena illustrée au a) de la figure **??**. Le centre de chaque cercle represente un point clé,la taille du rayon est proportionelle au σ du point clé,et le rayon déssiné indique l'orientation principale.

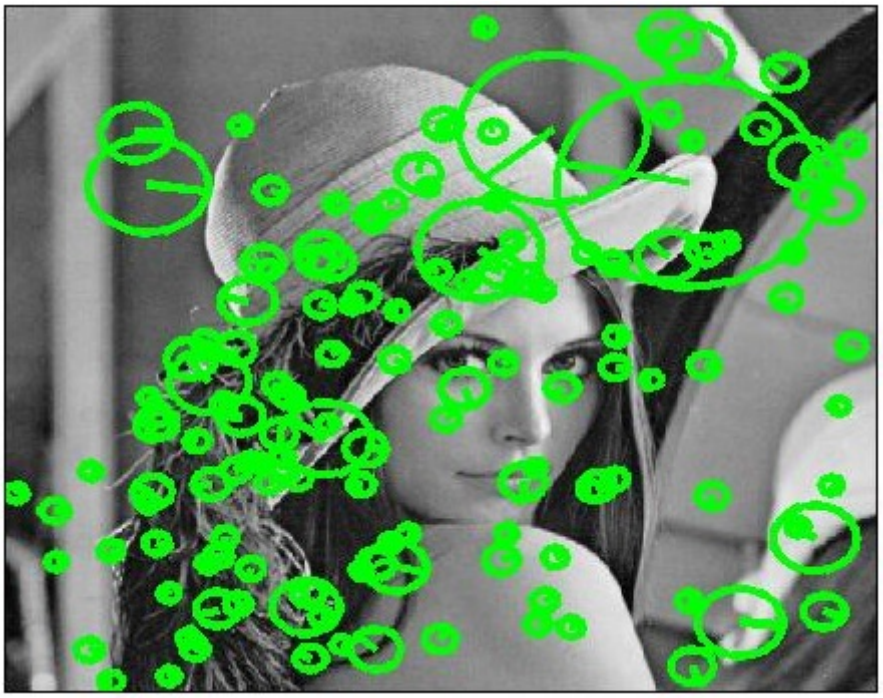

FIGURE 1.9: Illustrations de points clés

1.3.2.2 Descripteur SIFT du point d'intérêt

Après avoir désigné les points clés et leur orientation principale, il est maintenant temps de calculer le vecteur descripteur de chaque point clé. Tout comme l'étape précédente, le calcul qui suit s'effectue sur l'image lissée $L(x, y, \sigma)$ avec σ le plus proche du facteur d'échelle du point.

Pour chaque point, on commence par modifier le système de coordonnées local, en utilisant une rotation d'angle égal à l'orientation du point-clé, mais de sens opposé. On considère ensuite, toujours autour du point-clé, une région de 16×16 pixels, subdivisée en 4×4 zones de 4×4 pixels chacune. Sur chaque zone est calculé un histogramme des orientations comportant 8 intervalles.

En chaque point de la zone, l'orientation et l'amplitude du gradient sont calculés comme précédemment. L'orientation détermine l'intervalle à incrémenter dans l'histogramme, ce qui se fait avec, comme précédemment, une double pondération :Par l'amplitude et par une fenêtre gaussienne centrée sur le point clé, de paramètre égal à $0,5$ fois le facteur d'échelle du point-clé comme l'illustre la figure 1.10.

Ensuite, les 16 histogrammes à 8 intervalles chacun sont concaténés et normalisés. Dans le but de diminuer la sensibilité du descripteur aux changements de luminosité, les valeurs sont plafonnées à $0,2$ et l'histogramme est de nouveau normalisé, pour finalement fournir le descripteur SIFT du point-clé, de dimension 128.

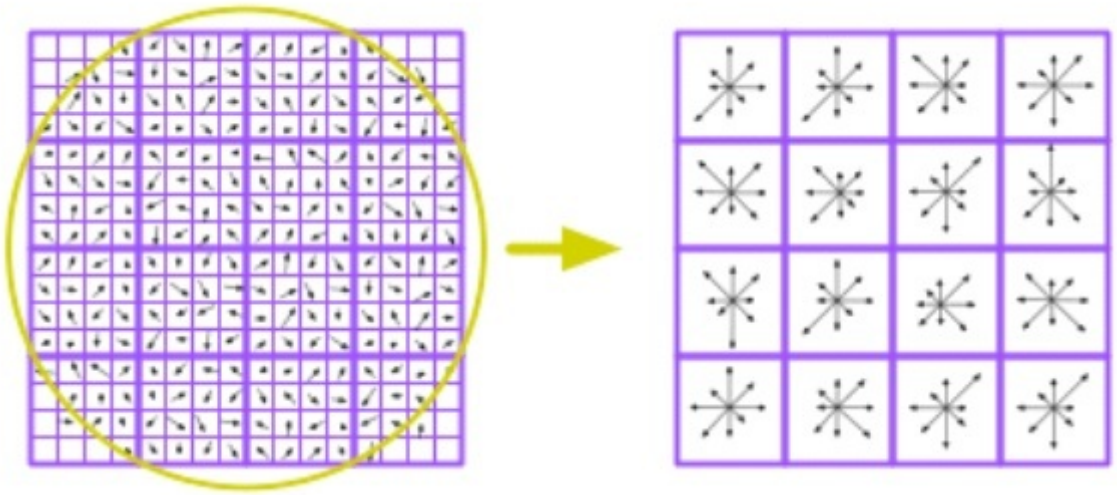

FIGURE 1.10: Calcul du descripteur d'un point clé

1.3.3 Correspondance entre images

La problématique de base pour laquelle la méthode SIFT a été conçue est la suivante : peut-on trouver dans une image donnée (image requête), des objets déjà présents dans une collection d'images de référence préétablie.
Afin de parvenir à cet objectif, il faut extraire de chaque image de la collection ces points clés et stocker les descripteurs. Ainsi au moment de la comparaison, il faut pour chaque point clé de l'image requête déterminer son plus proche voisin, en utilisant la distance Euclidienne, sur chaque image de la collection.
Afin d'éliminer les fausses correspondances, Lowe à utilisé un critère, qu'il a nommé critère d'unicité. Son énoncé est comme suit :

$$D_1 * s < D_2 \tag{1.16}$$

Avec
D_1 Distance avec le plus proche voisin.
D_2 Distance avec le deuxième plus proche voisin.
s Seuil de valeur supérieur à 1, fixé au préalable.
Une correspondace est prise en compte seulement si cette condition est vérifiée.Ce critère permet d'élliminer les fausses correspondances causées par des fonds d'images differents. Si le recourt a une recherche exhaustive du plus proche voisin est possible quand la base de reference est de taille raisonnable, Celà n'est plus envisageable quand la base est volumineuse. Pour pallier cela, Lowe propose d'untiliser une variante de l'algorithme de l'arbre Kd, appelé "Best-Bin-First"(BBF)

Conclusion

Le SIFT est un algorithme de reconnaissance d'objet. Il se base sur une approche locale pour la description des images à l'aide de points d'intérêt. Il permet d'extraire des points clés stables. Il est de ce fait robuste vis-à-vis des différentes variations de l'image.
Ce chapitre nous a permis de découvrir cet algorithme, et de nous familiariser avec les différents outils utilisés tels que filtres gaussiens.

Chapitre 2

SIFT : Implémentation

2.1 Introduction

Dans le chapitre précédent, nous avons découvert l'algorithme SIFT théoriquement. Nous avons aussi vu ses applications multiples.
Dans cette partie, nous présentons l'implémentation de la partie extraction des points d'intérêt sur Matlab. La dernière section du chapitre est consacrée aux résultats de l'implémentation.

2.2 Détails de l'implémentation

La figure 2.1 résume les étapes permettant l'extraction et le calcul de descripteurs SIFT. L'organigramme décrit les étapes suivies. Celles encadrées en bleu sont celles réalisées dans notre implémentation. Pour les étapes encadrées en orange, nous avons utilisé les fonctions écrites par Andrea Vedaldi [3]. Les codes que nous avons développés comportent quelques différences par rapport à ceux de Andrea Vedaldi :

- La taille du noyau pour le calcul de l'image filtrée est fixée à 7, et non pas proportionnelle à σ.
- Pour la réduction de la taille de l'image, lors du passage d'une octave à une autre , la fonction "imresize" de MATLAB a été utilisée. Celle-ci réduit les dimension par interpolation du type plus proche voisin.Vedaldi lui effectue un sous-échantillonnage de l'image sans interpolation.

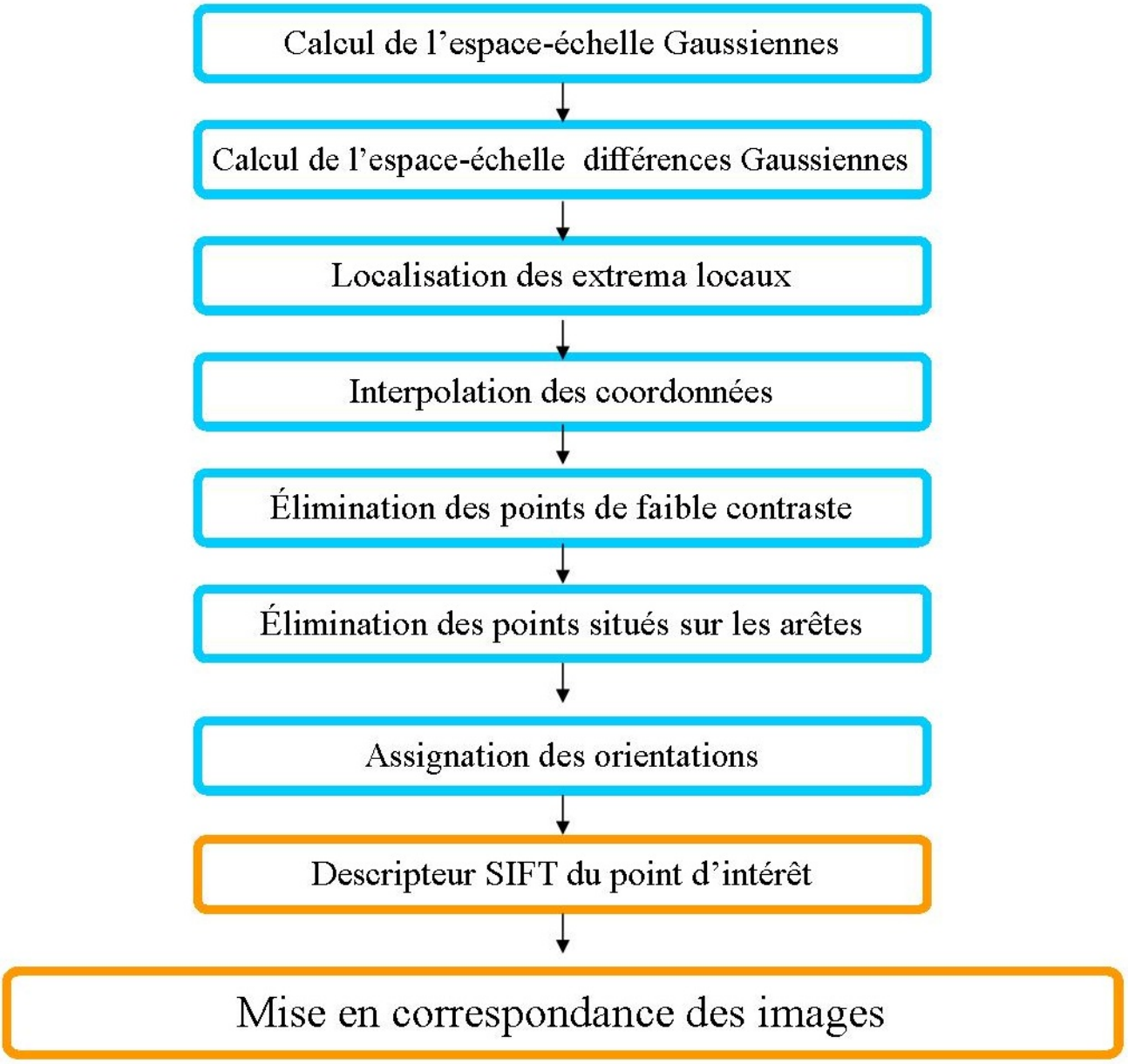

FIGURE 2.1: Etapes du SIFT

2.3 Evaluation de l'implémentation

2.3.1 Mise en correspondance de deux images

Mise en correspondance de deux images comportant le même contenu

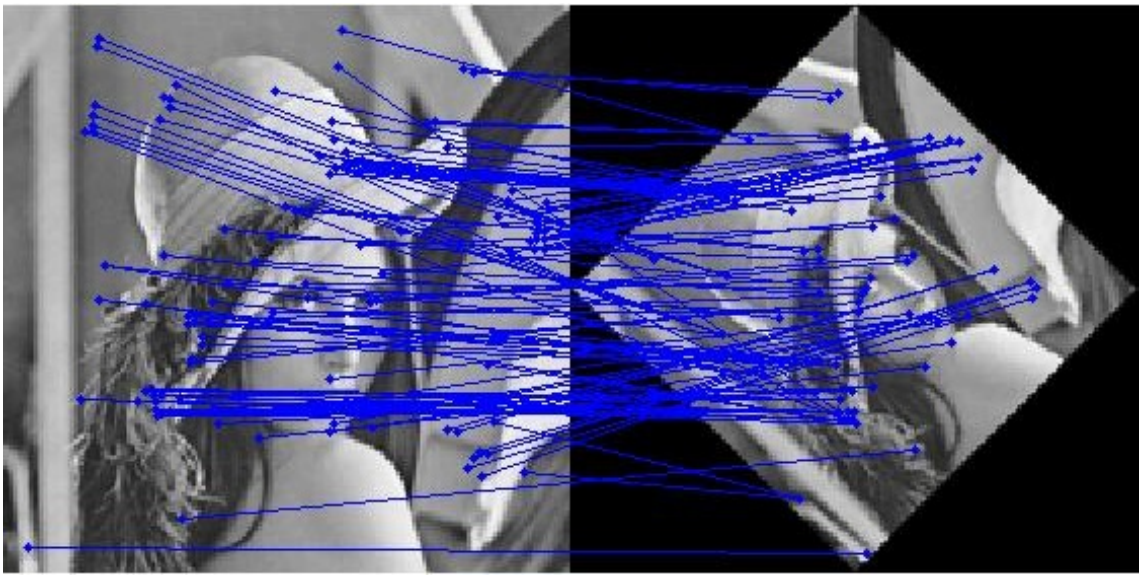

FIGURE 2.2: Mise en correspondance de deux images comportant le même objet

Mise en correspondance de deux images avec des contenus différents

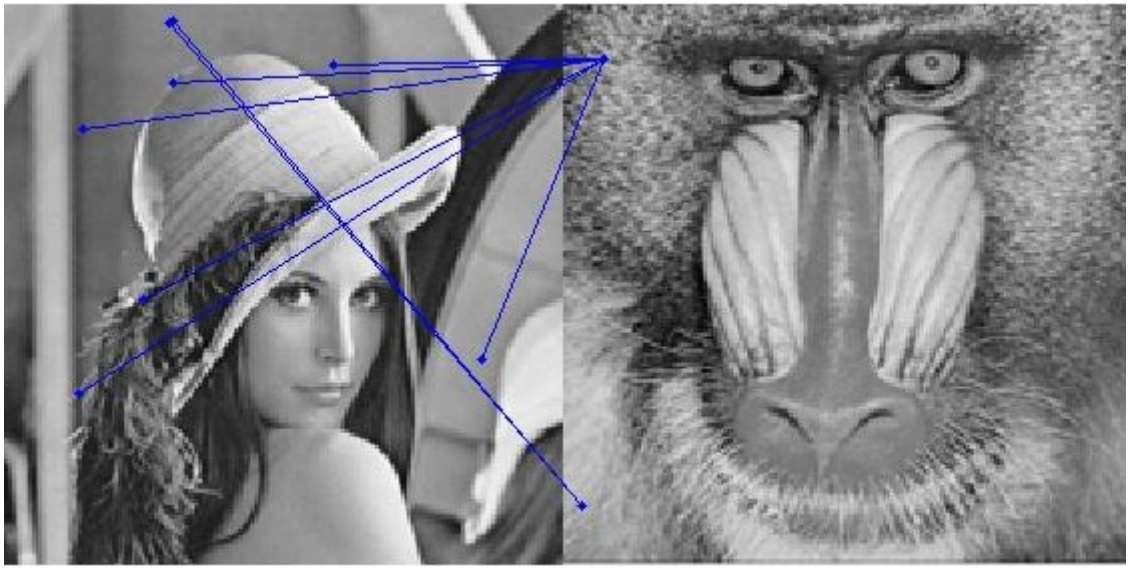

FIGURE 2.3: Mise en correspondance de deux images avec des contenus différents

2.3.2 Résultat sur la base de données d'image

Nous avons calculé les pourcentages de reconnaissance sans aucun changement effectué sur les images de la base de données(base décrite en annexe B). Nous présentons les résultats obtenus avec notre implémentation et avec celle de Vedaldi sont présentés dans le tableau 2.1.

	Top1	Top2	Top2
Implémentation de Vedaldi	91.667	96.875	98.958
Notre implémentation	95.833	98,958	98,958

TABLE 2.1: Poucentages de reconnaissance sans aucun changement

Nous remarquons que les pourcentages obtenus par notre implémentation sont légèrement supérieurs. Néanmoins l'implémentation de Veldadi est plus rapide.

Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'implémentation de la partie extraction des points d'intérêt sur Matlab.
Les résultats obtenus grâce à cette implémentation sont satisfaisants, avec plus de 95% de reconnaissance. Cependant, le temps d'exécution de l'algorithme sur Matlab reste important.

Conclusion générale

Dans ce projet, nous étudions un algorithme de traitement d'image, écrit spécifiquement pour la reconnaissance d'objet et la mise en correspondance des images.

Il s'agit d'un algorithme se basant sur une approche locale pour l'extraction et la description des points clés. Cet algorithme est censé être robuste par rapport aux variations de l'image comme les changements de luminosité, d'échelle ... etc.

Nous avons pu implémenter la partie extraction des points d'intérêt sur Matlab. Nous avons d'ailleurs obtenu des résultats très satisfaisants, bien que l'exécution du programme soit gourmande en temps.

Annexe A

Programmes SIFT sur Matlab

A.1 Fonction Principale

```
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%**************************************************************************
%********************  Soutenance de Master en Electronique*****************
%***************************  Programmes SIFT sous Matlab*********************
%*******************************  Nom: OUDNI  ********************************
%*******************************  Prénom: Louiza  ****************************
%*************************  Email: louiza.oudni@gmail.com  *******************
%**************************************************************************
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fonction Principale:
%Entrée:
%        I:       Image en niveau de gris.
%Sorties:
%       frames:  Matrice 4*K , avec K le nombre de points clés de l'image
%                dans chaque colonne: x,y,sigma,theta.
%       descri: Matrice 128*K , les descripteurs des points clés.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [frames,descri]=sift_ecrit(I)
%Calcul de l'espace-échelle Gaussiennes
L=Gscale(I);
%Calcul de l'espace-échelle  différences Gaussiennes
Dog = DoG(L);
%Localisation des extrema locaux
P = Localmax(Dog);
%Interpolation des coordonnées
%et Élimination des points de faible contraste
Pp=preciLocalmax(Dog,P);
%Assignation d'orientation
Ppts=orientationmax(L,Pp)
```

```
%Calcul du descripteur , en utilisant la fonction
% siftdescriptor écrite par Andrea Vedaldi.
descri=[];
frames=[];
for(i=1:length(Ppts))
     if(size(Ppts(i).P)~=[0,0])
sh = siftdescriptor(...
      L(i).img, ...
      Ppts(i).P, ...
      1.6*2^(1.3), ...
      3, ...
      -1, ...
      'Magnif', 3, ...
      'NumSpatialBins', 4, ...
      'NumOrientBins', 8) ;
  Ppts(1).P(1:2,:)=Ppts(1).P(1:2,:)*2^(i-2);
  frames=[frames,Ppts(i).P]
  descri=[descri,sh];
     end
end
```

A.2 Calcul de l'espace-échelle Gaussiennes

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Calcul de l'espace-échelle Gaussiennes:
%Entrée:
%        img:       Image en niveau de gris.
%Sortie
%          L:        Espace-échelle Gaussiennes.
%                    C'est une structure, pou accéder a l'image i
%                     de l'octave j :  L(j).img(:,:,i) .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function L = Gscale(img)
%Fixer :sigma0, taille du noyau, facteur k,
% et nombre d'octaves (levels)
sigma0=1.6;
gsize=[7 7];
[a b]=size(img);
img=double(img);
levels=floor(log2(min(a,b)))-2;
facteur=2;
k=2^(1/3);
%Pour chaque Octave
for i = 1:levels
        sigma=sigma0*2^(i-2)
        %Image a la bonne taille:
        image=imresize(img,facteur);
         facteur=facteur/2;
     for(j=1:6)
         %Calcul du noyau du filtre Gaussien
        g = fspecial('gaussian',gsize,sigma);
        %Filter l'image
        im = imfilter(image,g,'conv');
```

```
        %La sauvegarder
        L(i).img(:,:,j) = im;
        L(i).sigma(j)=sigma

         sigma=sigma*k
     end
end
end
```

A.3 Calcul de l'espace-échelle différences Gaussiennes

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Calcul de l'espace-échelle  différences Gaussiennes
%Entrée:
%       L:       Espace-échelle Gaussiennes.
%Sorties:
%          Dog:       Espace-échelle différences Gaussiennes.
%                    C'est une structure, pou accéder a l'image i.
%                     de l'octave j :  L(j).img(:,:,i) .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Dog = DoG(L)
%Pour chaque octave
for i = 1:length(L)
     for(j=1:size(L(i).img,3)-1)
         %Pour chaque image j : soustraction (image j+1)-(image j)
         Dog(i).img(:,:,j)= L(i).img(:,:,j+1)-L(i).img(:,:,j);
         Dog(i).sigma(j)=L(i).sigma(j)
     end
end
end
```

A.4 Localisation des extrema locaux

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Localisation des extrema locaux
%Entrée:
%       Dog:      Espace-échelle différences Gaussiennes.
%Sortie:
%       P:      maxima et minima du Dog.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function P = Localmax(Dog)
sigma0=1.6;
k=2^(1/3)
%Pour chque octave
for i = 1:length(Dog)
    sigma=sigma0*2^(i-2);

    P(i)=struct('x',[],'y',[],'s',[],'sig',[]);
    %Recherche des maxima:
```

```
[r2, c2, s2] = ind2sub(size(Dog(i).img), find(imregionalmax(Dog(i).img)));
    %Recherche des mimma:
[r1, c1, s1] = ind2sub(size(Dog(i).img), find(imregionalmax(-Dog(i).img)));
    %Concatenation des deux:
     r=[r1;r2];
     c=[c1;c2];
     s=[s1;s2];
 sig=sigma*k.^(s-1);
    %Eliminer ceux trop proches des bords:
 border = 10;
valid = r - 1 >= border & size(Dog(i).img, 1) - r >= border & ...
        c - 1 >= border & size(Dog(i).img, 2) - c >= border & ...
        s - 1 >= 1      & size(Dog(i).img, 3) - s >= 1;
[r, c, s, sig] = deal(r(valid), c(valid), s(valid), sig(valid));
 P(i).x=r;
 P(i).y=c;
 P(i).s=s;
 P(i).sig=sig;
 clear r c s sig
end

end
```

A.5 Interpolation des coordonnées et élimination des points de faible contraste

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Interpolation des coordonnées
                    % et Élimination des points de faible contraste
%Entrées:
%       Dog:      Espace-échelle différences Gaussiennes
%       P:        maxima et minima du Dog
%Sortie:
%       Pp:       Points interpolés
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Pp=preciLocalmax(Dog,P)

compt=0;
%Pour chaque octave:
    for i = 1:length(Dog)

     Img=double(Dog(i).img);
     [dim1 dim2]=size(Img(:,:,1));
     for j=1:length(P(i).x)
         x=P(i).x(j);
         y=P(i).y(j);
         s=P(i).s(j);
         condition=0;
        while condition==0
           if((x>1) & (y>1) & ((dim1-x)>2) & ((dim2-y)>2) & s>1  & s<5)
```

```
% Calcul du gradient.
                dim2
                Dx = 0.5 * (Img(x+1,y,s) - Img(x-1,y,s));
                Dy = 0.5 * (Img(x,y+1,s) - Img(x,y-1,s));
                Ds = 0.5 * (Img(x,y,s+1) - Img(x,y,s-1)) ;
% Calcul de la Hessienne.
                Dxx = (Img(x+1,y,s) + Img(x-1,y,s) - 2.0 * Img(x,y,s)) ;
                Dyy = (Img(x,y+1,s) + Img(x,y-1,s) - 2.0 * Img(x,y,s)) ;
                Dss = (Img(x,y,s+1) + Img(x,y,s-1) - 2.0 * Img(x,y,s)) ;
                Dxy = 0.25 * ( Img(x+1,y+1,s) + Img(x-1,y-1,s) - Img(x-1,y+1,s) - Img(x+1,y-1,s) );
                Dxs = 0.25 * ( Img(x+1,y,s+1) + Img(x-1,y,s-1) - Img(x-1,y,s+1) - Img(x+1,y,s-1) );
                Dys = 0.25 * ( Img(x+0,y+1,s+1) + Img(x,y-1,s-1) - Img(x,y-1,s+1) - Img(x,y+1,s-1) );


                H=[Dxx,Dxy,Dxs;...
                    Dxy,Dyy,Dys;...
                    Dxs,Dxy,Dss];

                G=[Dx;Dy;Ds];
                %Calcul de x chapeau
                preci=-inv(H)*G;
                z=[x ;y ;s];
                %Si x chapeau>0.5
                [a b]=find(abs(preci>0.5));
                if(size(a)~=[0 0])
                       preci;
                       z;
                      z(a)=z(a)+sign(preci(a)).*ceil(abs(preci(a)));
                      compt=compt+1;
                      if((z(1)>1) & (z(2)>1) & ((dim1-z(1))>1) & ((dim2-z(2))>1) & z(3)>1  & z(3)<5)
                      x=z(1);y=z(2);s=z(3);
                      condition=0
                      else
                      condition=1
                      end
                else
                    condition=1;
                end
         else
            condition=1;
         end
        end

         D=Img(x,y,s)+(0.5)*G'*preci
         %Élimination des points de faible contraste
         rapport=(Dxx+Dyy)^2/(Dxx*Dyy-Dxy*Dxy);
         r=(10+1)^2/10;
         if(abs(D)>0.03  & rapport<r & rapport>=0)
              Pp(i).x(d)=x+preci(1);
              Pp(i).y(d)=y+preci(2);
              Pp(i).s(d)=s+preci(3);
         d=d+1;

     end
    end
    end
```

```
end
```

A.6 Assignation d'orientation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Assignation d'orientation

%Entrées:
%       L:         Espace-échelle Gaussiennes
%       P:         maxima et minima du Dog
%Sortie:
%       Ptp:     Points clé avec leurs orientations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Pts=orientationmax(L,Pp)
compt=0;
%Pour chaque point clé
    for i = 1:length(Pp)
        d=0;
        Pts(i)=struct('P',[]);
        for j=1:length(Pp(i).x)
             Por(i)=struct('x',[],'y',[],'s',[],'theta',[]);
            H=zeros(1,36)
            xv=Pp(i).x(j);
            yv=Pp(i).y(j);

            x=floor(Pp(i).x(j)+0.5);
            y=floor(Pp(i).y(j)+0.5);
            s=floor(Pp(i).s(j)+0.5);
           if(s>0 & s<7)

            sigmaw=1.5*1.6* 2^(s/3);
            W=floor(3*1.6*2^(1/6));


            Img=L(i).img(:,:,s);
            [M N]=size(Img);
            if(x>8 & y>8 & M-x>8 & N-y>8)
            Img=double(Img);

            for  xs=max(-W,2-x):min(+W, M-2-x)
                for ys=max(-W, 2-y):min(+W, N-2-y)
                %Calcul du module et de l'angle
                Dx = 0.5 * ( Img(x+xs+1,y+ys) - Img(x+xs-1,y+ys) ) ;
                Dy = 0.5 * ( Img(x+xs,y+ys+1) - Img(x+xs,y+ys-1) ) ;
                dx = (x+xs) - xv;
                dy = (y+ys) - yv;

                win = exp( - (dx*dx + dy*dy)/(2*sigmaw*sigmaw) );
                modul = (Dx*Dx + Dy*Dy)^(1/2) ;
                theta = atan2(Dy, Dx) ;
                if(theta<0) theta=theta+2*pi
                end
                bin =int8(36*theta/(2*pi))+1 ;
```

```
                if(bin>36) bin=mod(bin,36)+1;
                end
                H(bin) =H(bin)+ modul*win ;
                 end
            end
            A=find(H>0.80*max(H));
            P=[yv xv s]';
            Pt=repmat(P,1,length(A));
            Pt=[Pt ; A*(2*pi/36)];
            Pts(i).P=[Pts(i).P ,Pt];
            if(i==2)
                tt=44
            end
            end
           end
    end
end
```

Bibliographie

[1] Maïtine Bergounioux. Quelques méthodes de filtrage en Traitement d'Image. Cours donné dans le cadre d'une école CIMPA - en attente de publication dans les actes. URL http://hal.archives-ouvertes.fr/hal-00512280.

[2] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2) :91–110, November 2004. ISSN 0920-5691. doi : 10.1023/B:VISI.0000029664.99615.94. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.