Jesús Bujalance Martín Encadrante: Florence Tupin

SEGMENTATION D'IMAGE SAR MULTI-TEMPORELLE

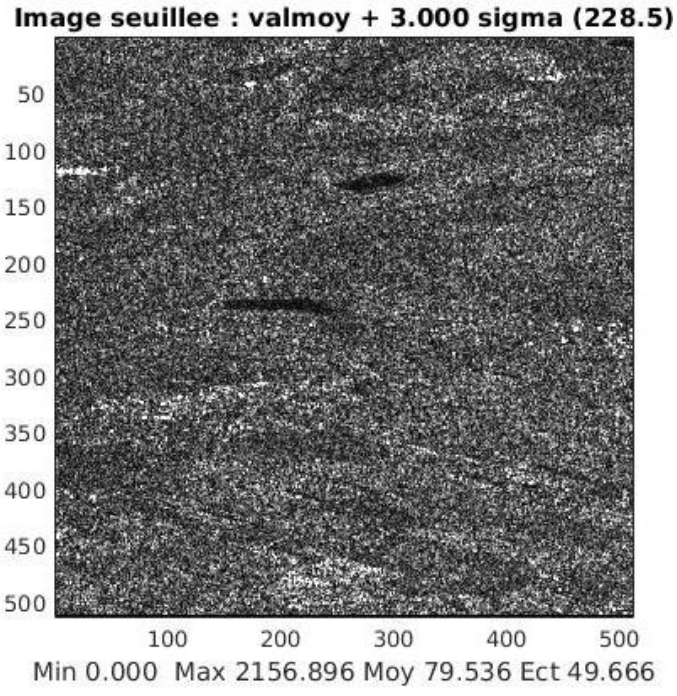
I. Introduction

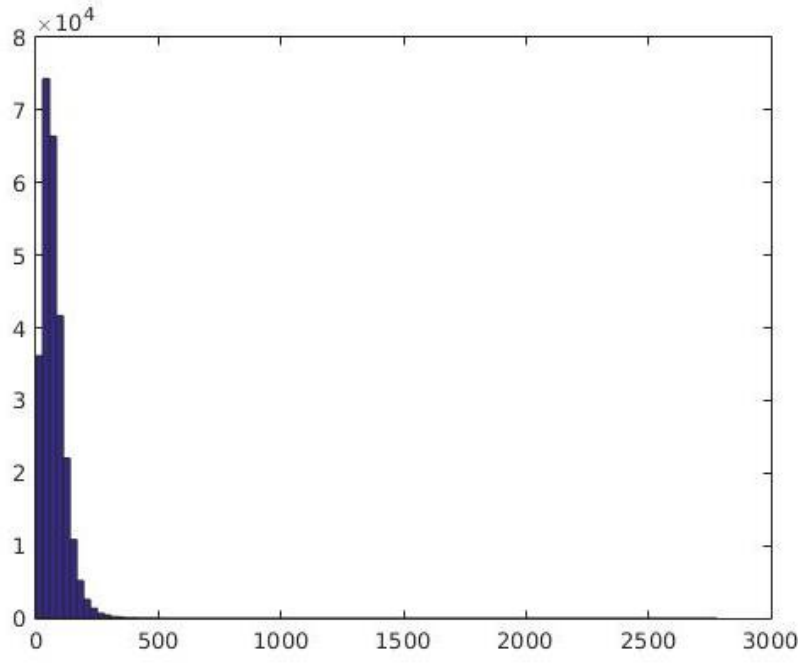
Il existe de nombreux algorithmes de segmentation qui permettent de segmenter une image dans des superpixels, qui doivent être cohérents avec les bords de l'image et d'autres aspects comme la couleur.

Un des derniers algorithmes apparus est SLIC, qui est basé sur l'algorithme k-means et donne de très bons résultats. Nous allons voir si cet algorithme est capable de segmenter correctement des images très bruitées comme les images SAR.

II. Contexte

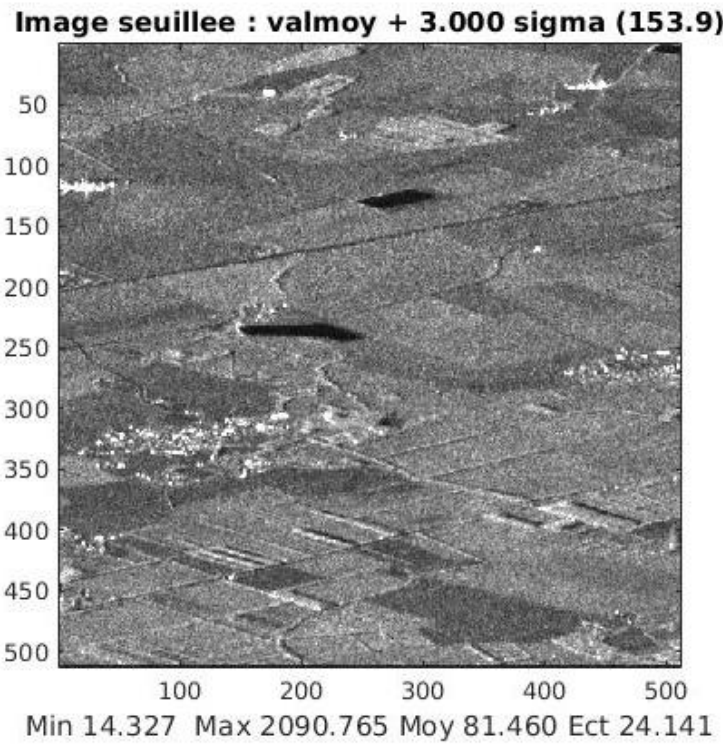
Nous disposons de 24 images SAR d'un même endroit à Chenguang prises à des moments différents. Voici une des images et son histogramme :



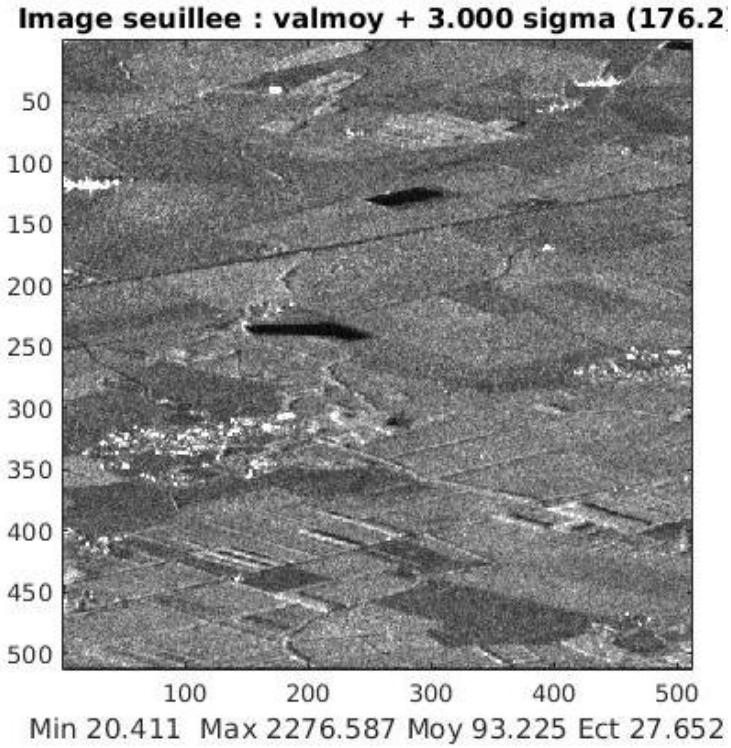


Ce sont des images avec des coefficients double, qui peuvent être très grands à cause du bruit. Pour les visualiser on utilise donc les fonctions *imz2mat.m* (pour transformer la pile d'images en une matrice) et *visusar.m* (qui fait un seuillage à partir de valmoy + 3*ecart-type). La documentation de ces fonctions se trouve dans ce lien : http://perso.telecom-paristech.fr/~nicolas/TIILAB/index.html

Si on réalise une moyenne temporelle de ces 24 images on obtient ça :

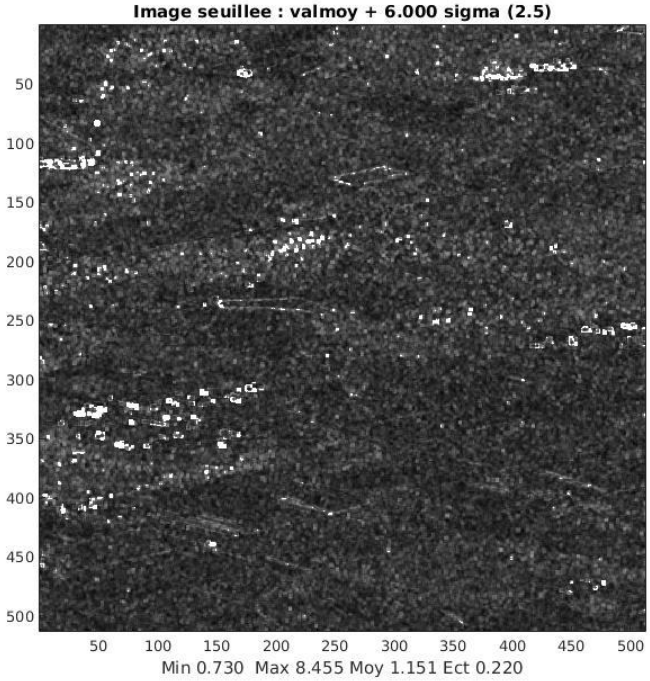


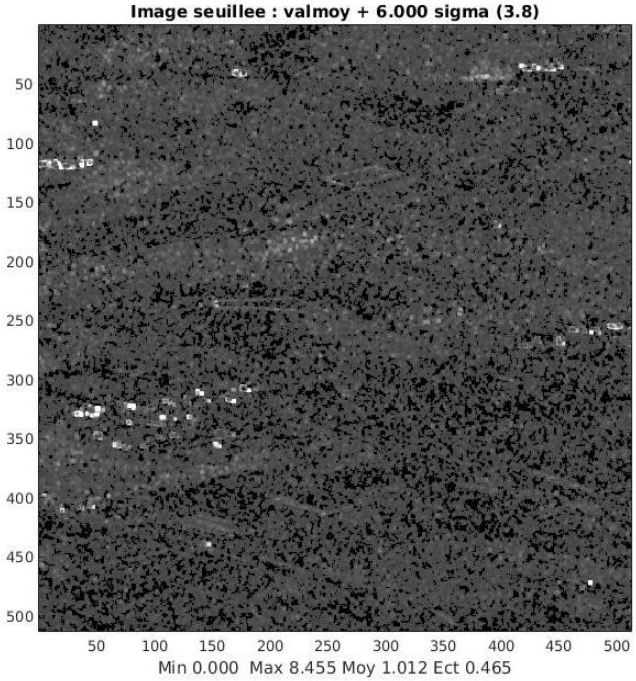
Moyenne L1



Moyenne L2

Toutes les opérations qui suivent ont été réalisés sur la moyenne temporelle L2.

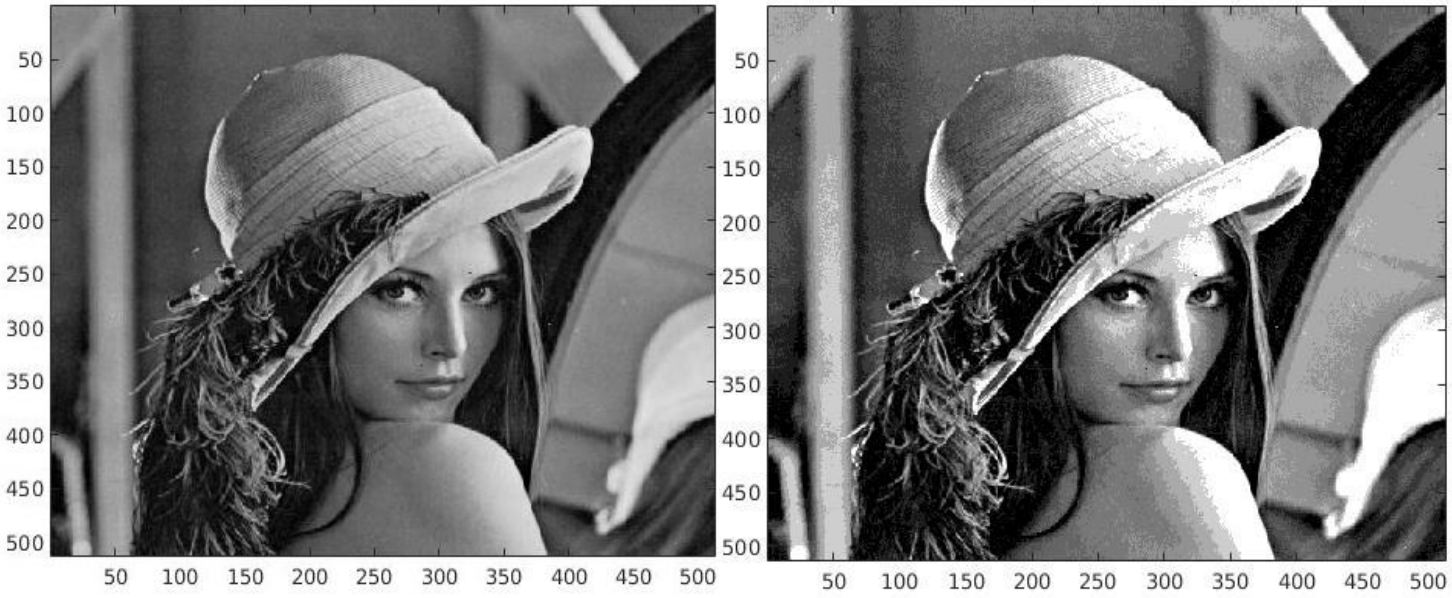




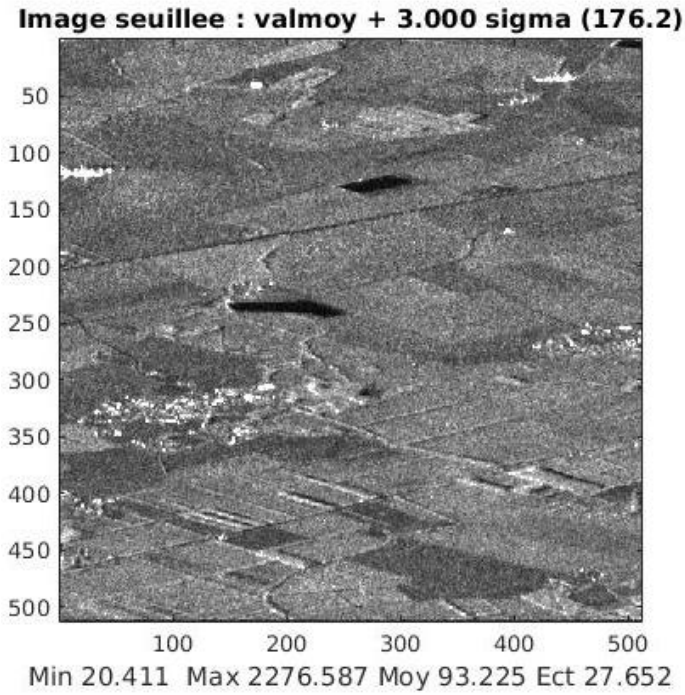
Pour mettre en évidence le caractère multi-temporel des images, nous pouvons regarder ces deux images. À gauche, chaque pixel représente le rapport écart-type sur moyenne de l'intensité des 24 pixels multi-temporels. Comme 24 n'est pas une quantité assez grande, nous faisons aussi une moyenne spatiale dans une fenêtre 3x3, donc finalement c'est le rapport écart-type sur moyenne de l'intensité de 24x9 pixels. A droite c'est la même image, mais les pixels avec une valeur < 1 ont été mis à 0 (ce sont les endroits où il n'y a eu pratiquement aucune variation temporelle entre les 24 images). Nous pouvons voir que la plupart des modifications temporelles sont à cause du bruit, les seules grandes variations se trouvent surtout au niveau des lumières. Les faisceaux des lumières peuvent en effet présenter des grandes différences d'un moment à un autre car ils sont très influencés par la météo notamment.

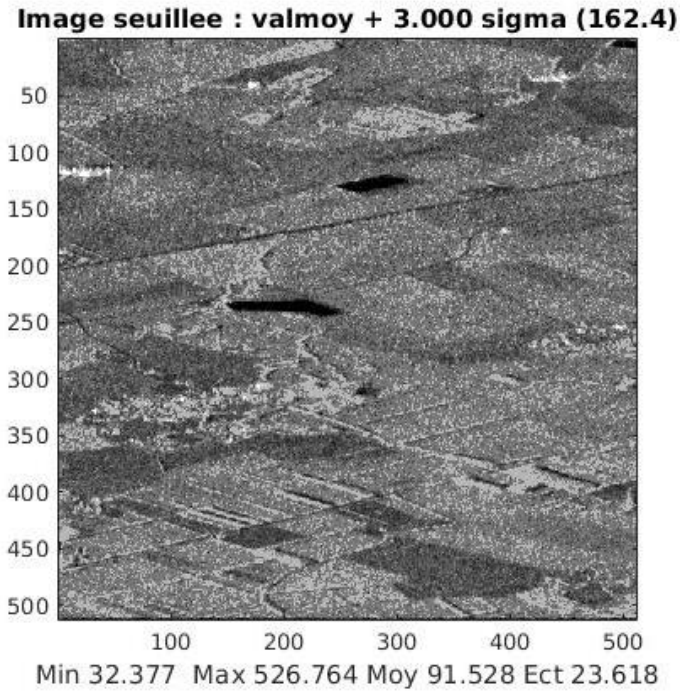
III. K-means

Nous avons implémenté l'algorithme de segmentation basique k-means. Pour l'initialisation nous avons reparti les K classes à des intervalles réguliers pour que chaque classe contienne le même nombre de pixels à l'initialisation. Le code complet se trouve en annexe.

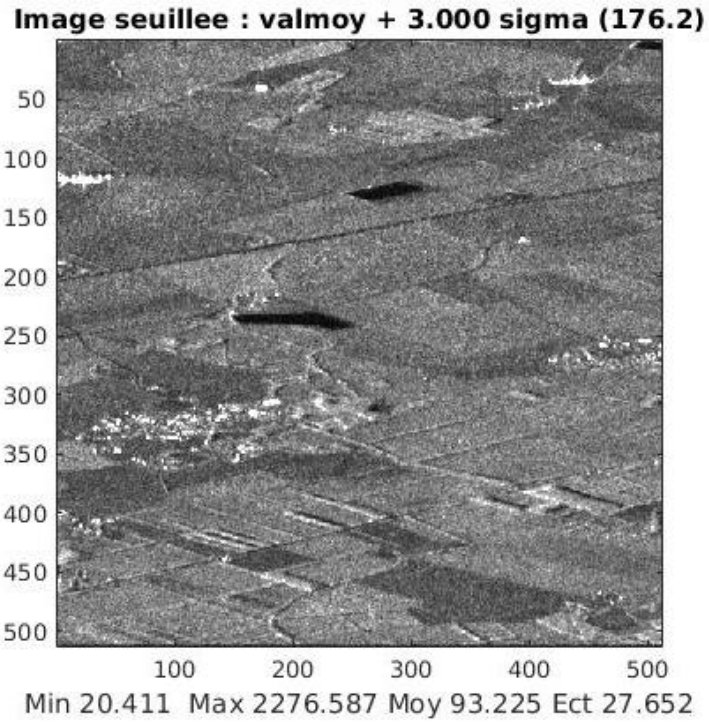


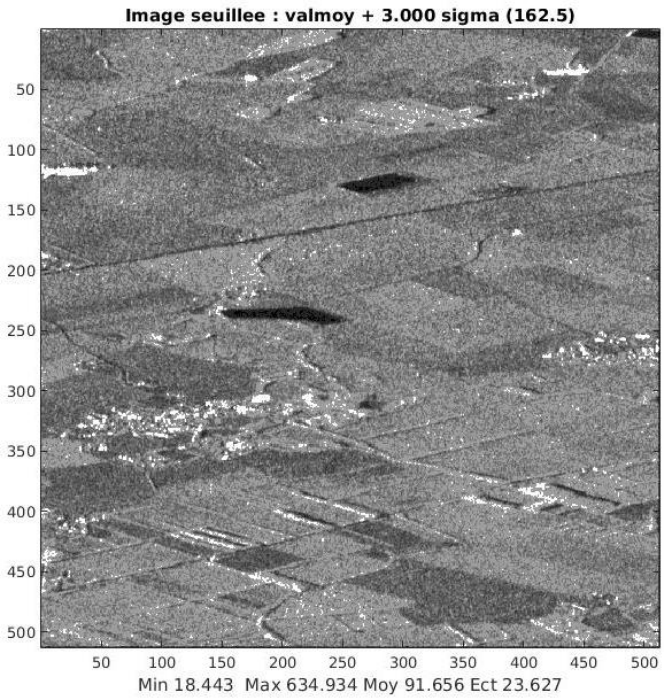
K=10





K=10





K=20

IV. SLIC

Nous avons implémenté une version basique de l'algorithme de segmentation SLIC. Le code complet se trouve en annexe. Nous nous sommes aidés ce cet article : https://infoscience.epfl.ch/record/177415

Voici la liste de simplifications qu'on a fait :

- Pour l'initialisation nous avons reparti les K centres des superpixels dans une grille rectangulaire à des intervalles réguliers. Cependant dans la plupart des implémentations qu'on peut trouver sur Internet, le choix est plutôt celui d'une grille hexagonale. Avec une telle grille on travaille avec des objets en 6-connexité, ce qui facilite les tâches de post-processing (combiner les superpixels)

- Pour l'initialisation, nous n'avons pas déplacé les centres de superpixels dans le minimum du gradient dans un voisinage 3x3. L'intérêt de le faire est d'éviter centrer un superpixel dans un bord ou un pixel bruité.

- Nous n'avons pas implémenté les étapes de post-processing, qui servent à assurer la connexité des superpixels en les combinant.

- Pour le calcul de la distance $D' = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2}$ où Ns est la distance spatiale maximale dans un superpixel et Nc la distance en couleur maximale dans un superpixel, nous avons consideré Nc comme constant, et Ns=S l'espacement entre les centres des superpixels dans la grille, pour obtenir D. Cette aproximation est suggerée par l'article et implémentée systématiquement en pratique.

$$D = \sqrt{{d_c}^2 + \left(\frac{d_s}{S}\right)^2 m^2}.$$

m=30, 225 superpixels

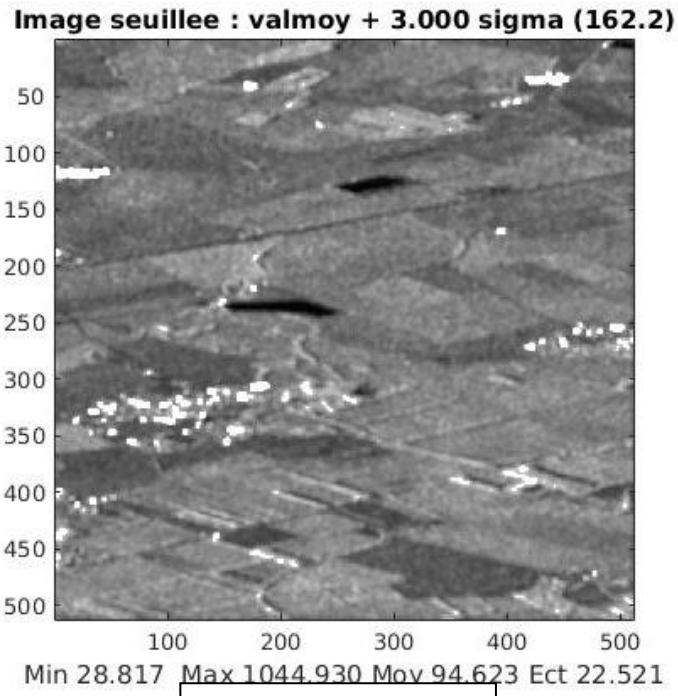
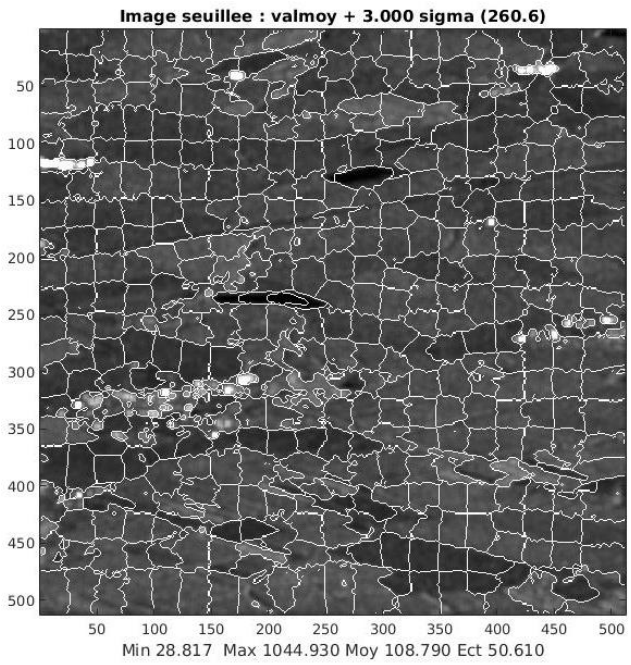
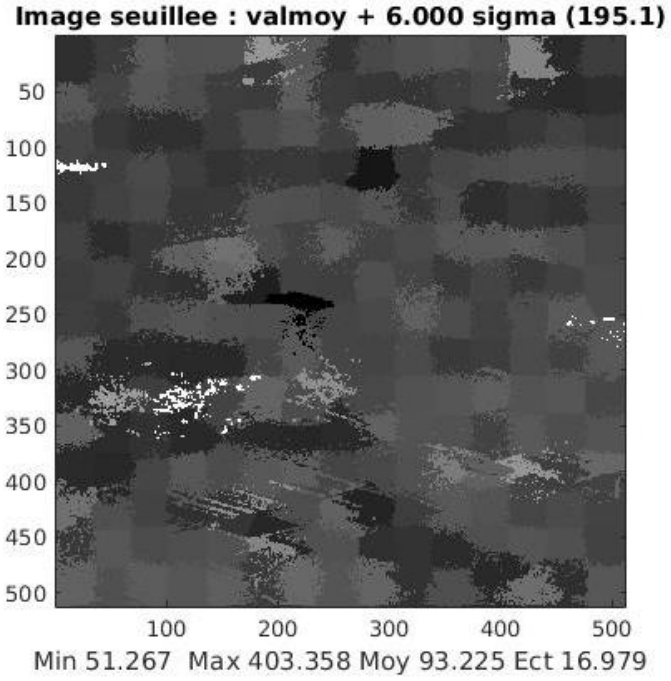


Image moyennée 5x5

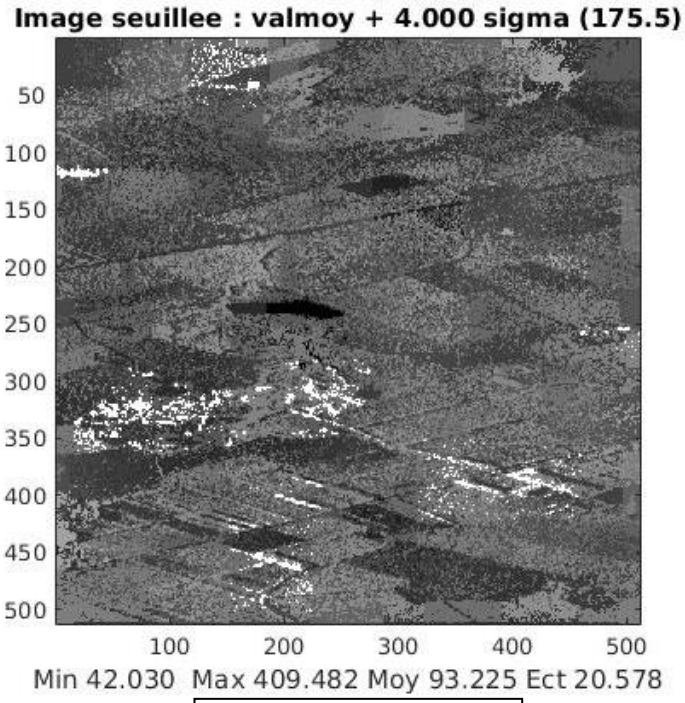


m=20, 400 superpixels

Le paramètre m détermine si on donne plus d'importance à la proximité spatiale ou en couleur. Lorsque m est grand, la proximité spatiale est plus importante et on obtient des superpixels plus compacts. Lorsque m est petit les superpixels adhèrent mieux aux bords de l'image mais ont des tailles moins et formes moins régulières.



m=40, 225 superpixels



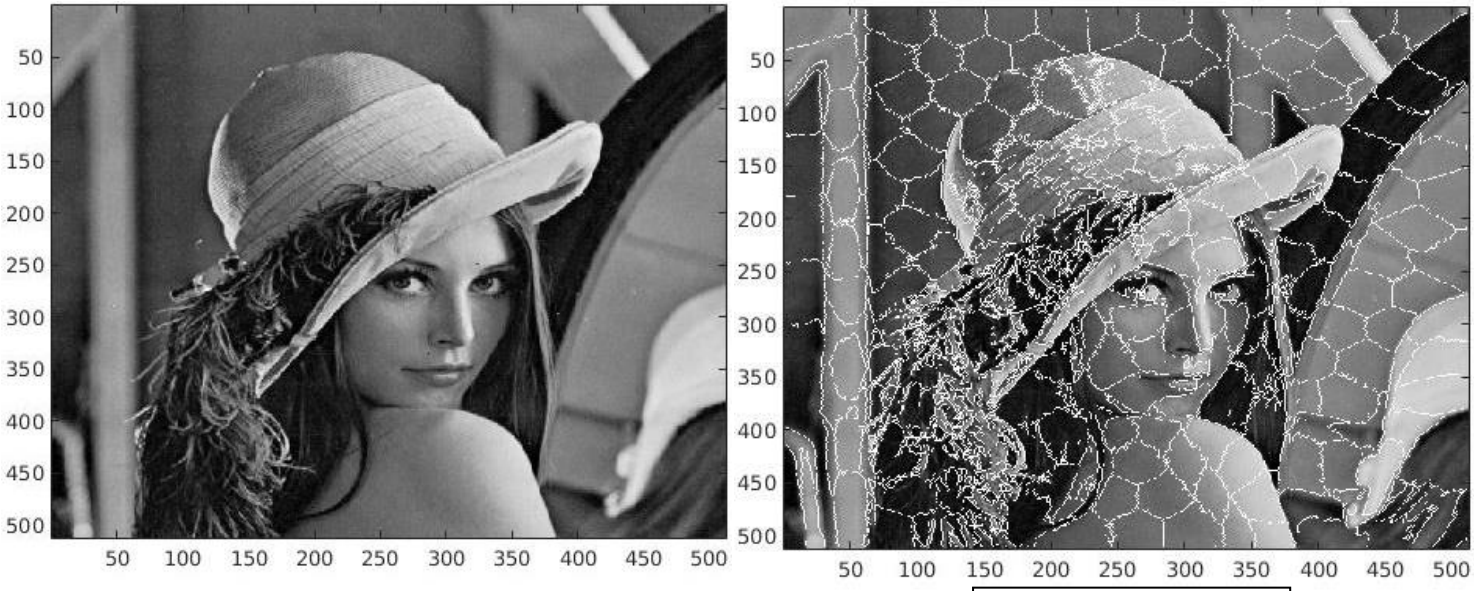
m=10, 225 superpixels

V. Résultats expérimentaux

Pour cette partie nous avons utilisé le code de SLIC de Peter Kovesi, qu'on peut trouver sur ce lien :

http://www.peterkovesi.com/projects/segmentation/

Nous l'avons adapté pour pouvoir l'utiliser avec des images en niveaux de gris.



m=30, 200 superpixels

Pour obtenir des meilleurs résultats, nous avons moyenné spatialement notre image pour filtrer le bruit.

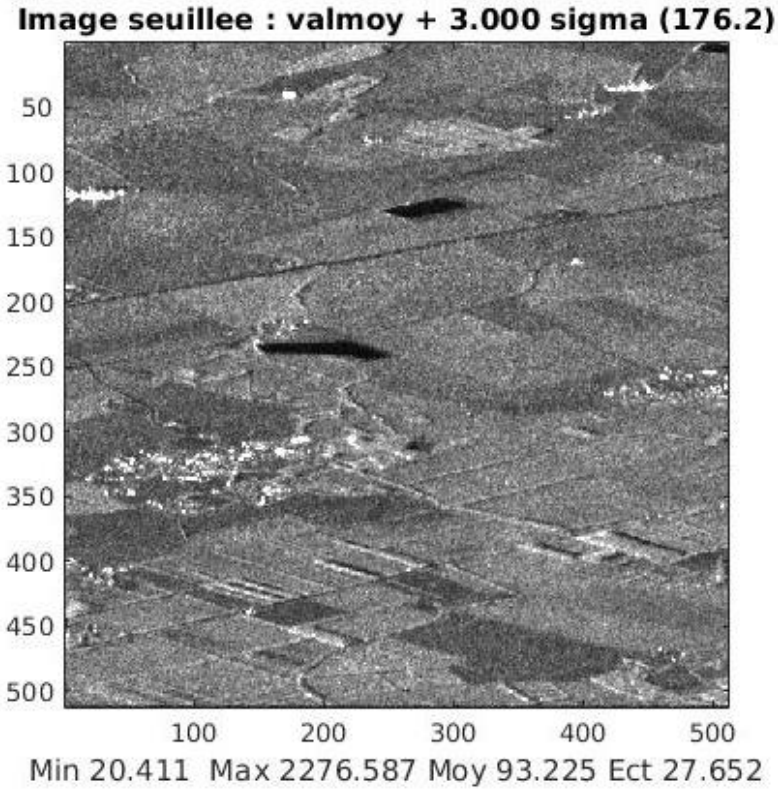
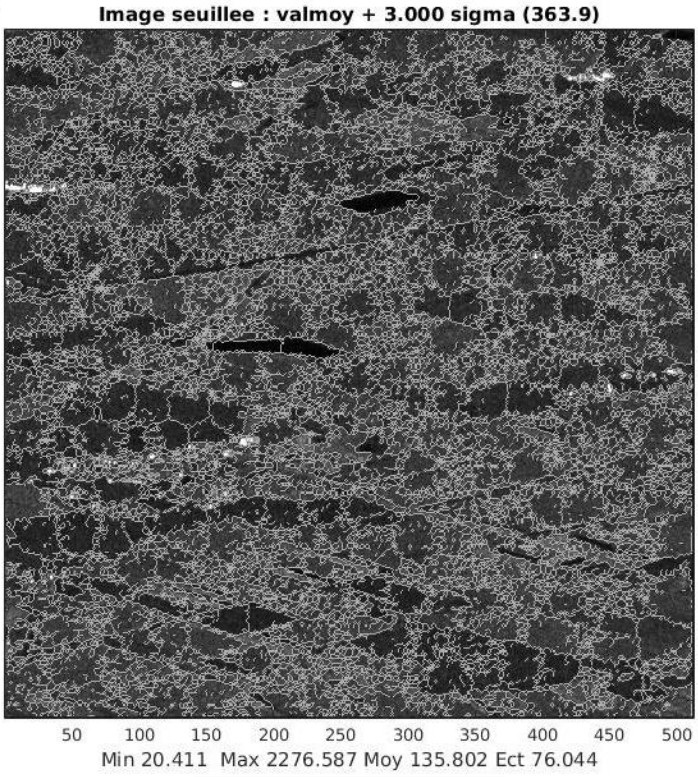


Image non filtrée



m=20, 400 superpixels

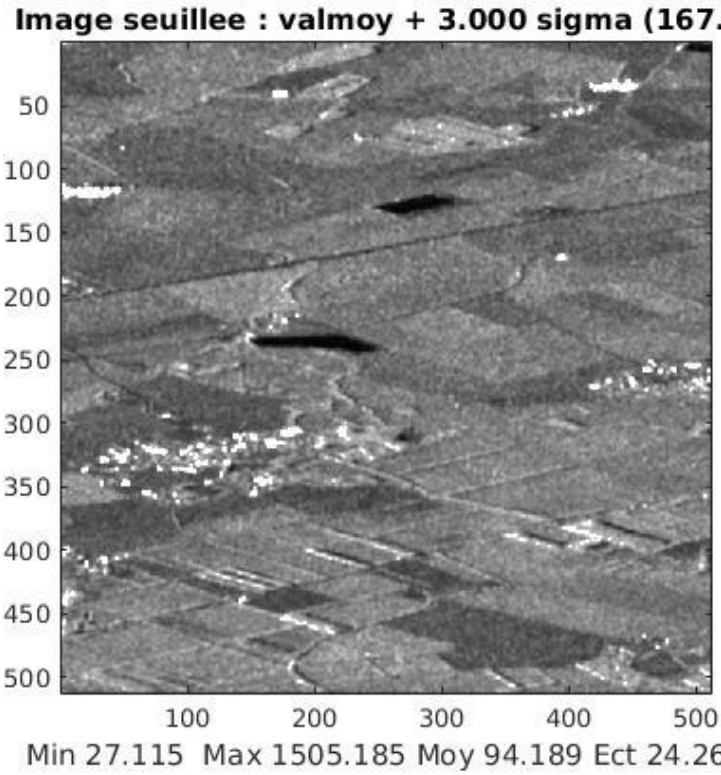
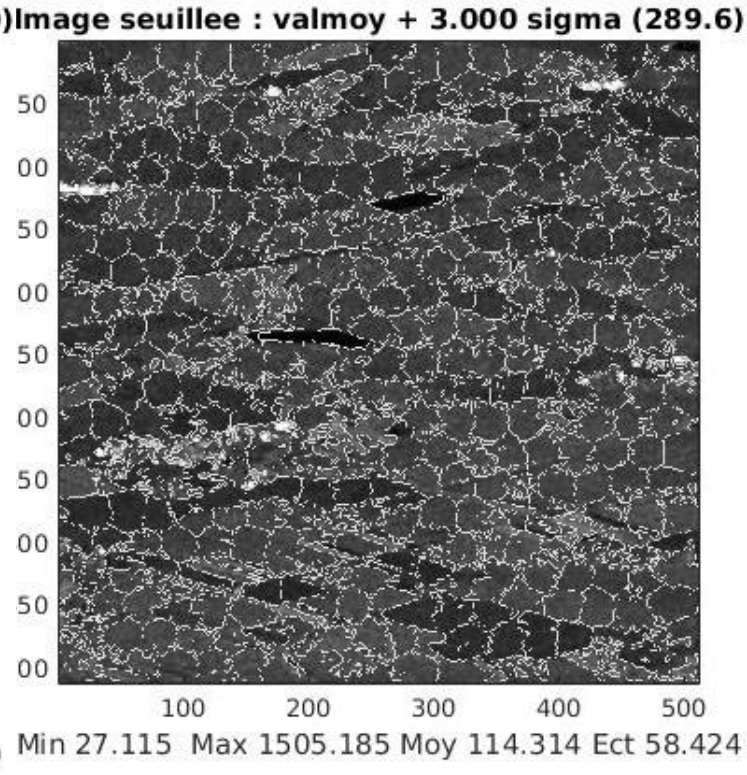


Image moyennée 3x3



m=20, 400 superpixels

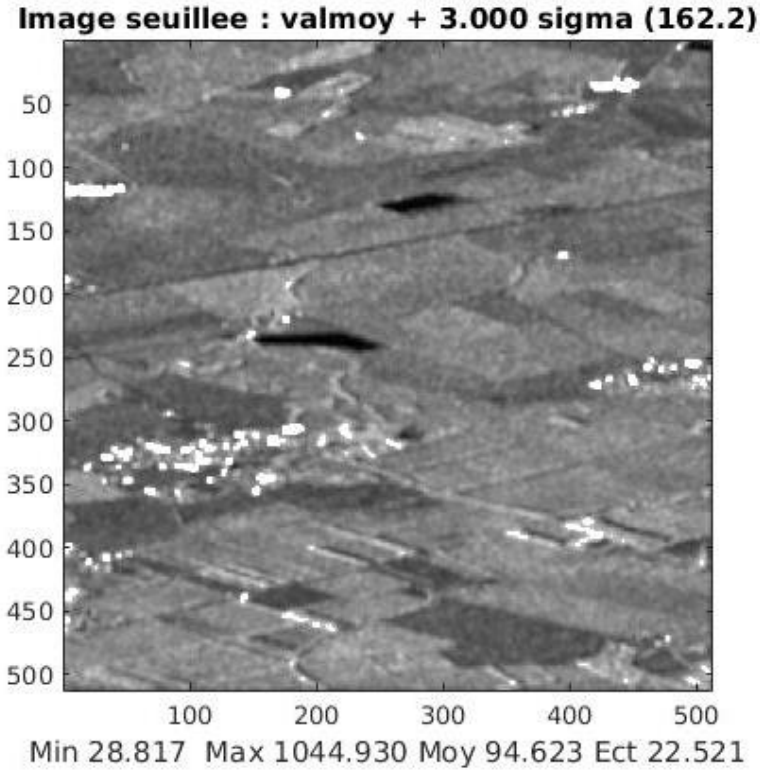
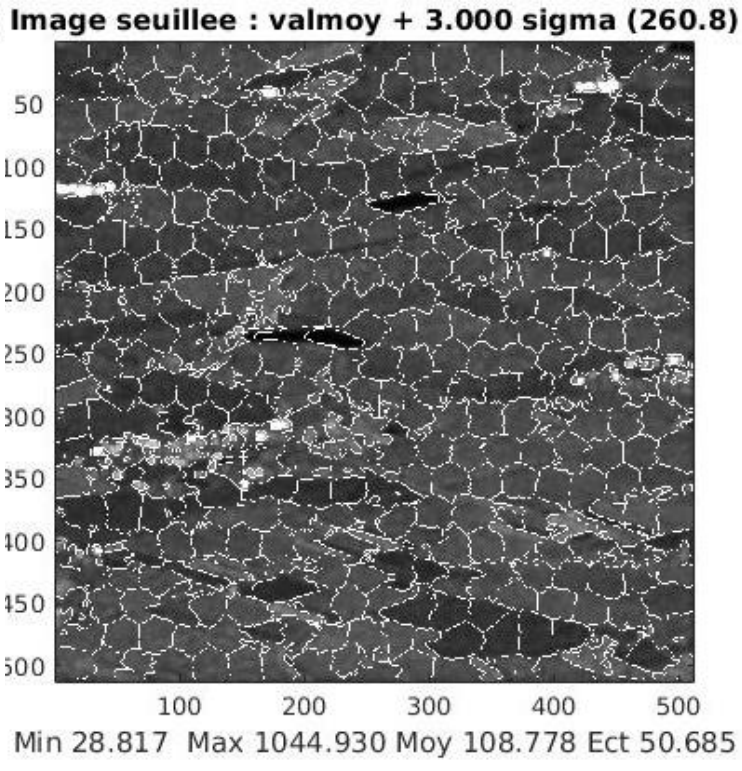


Image moyennée 5x5



m=20, 400 superpixels

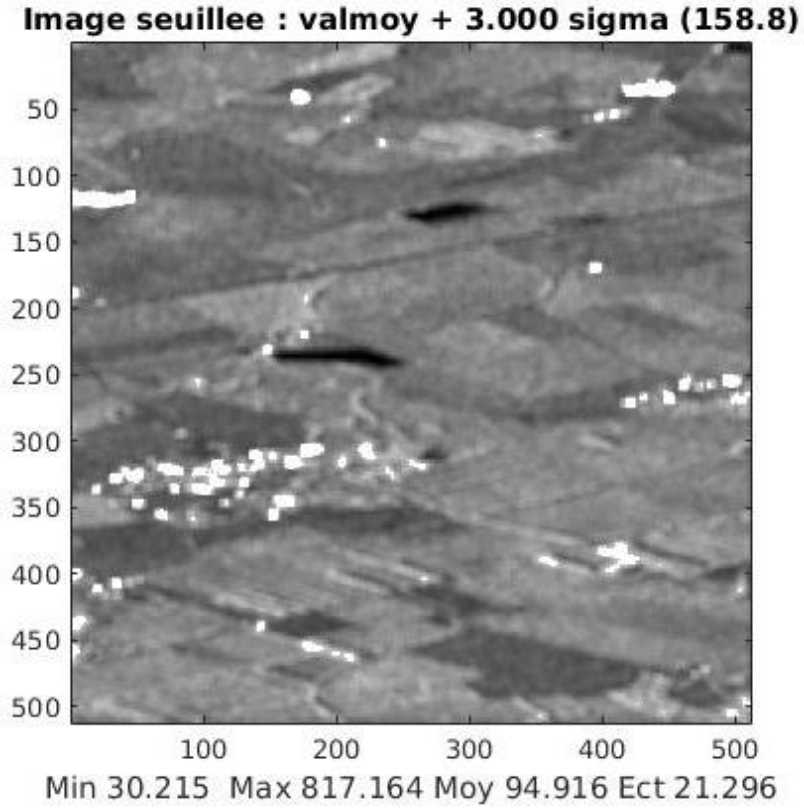
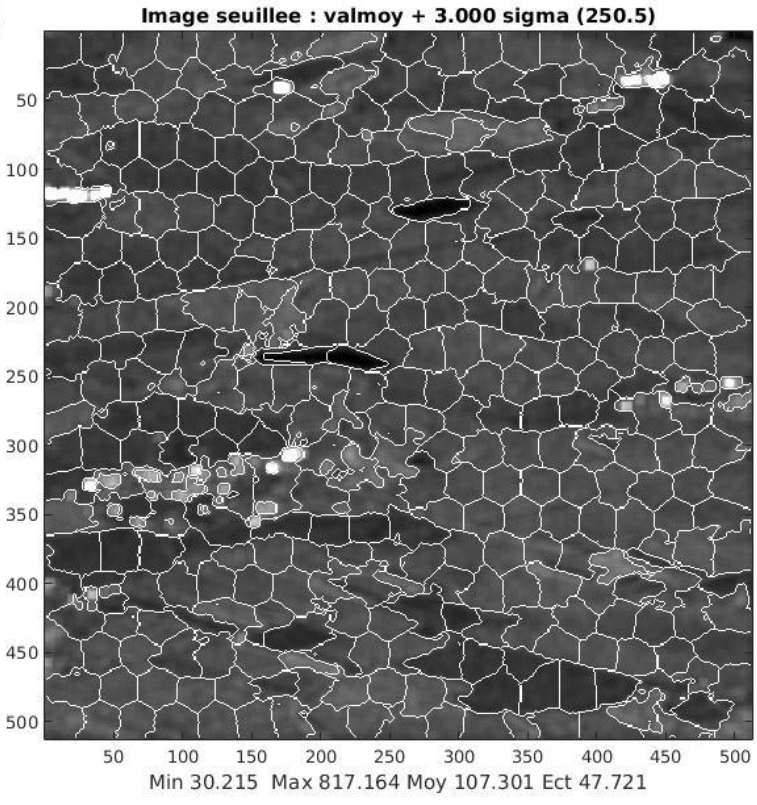


Image moyennée 7x7



m=20, 400 superpixels

Etudions maintenant l'influence du paramètre m

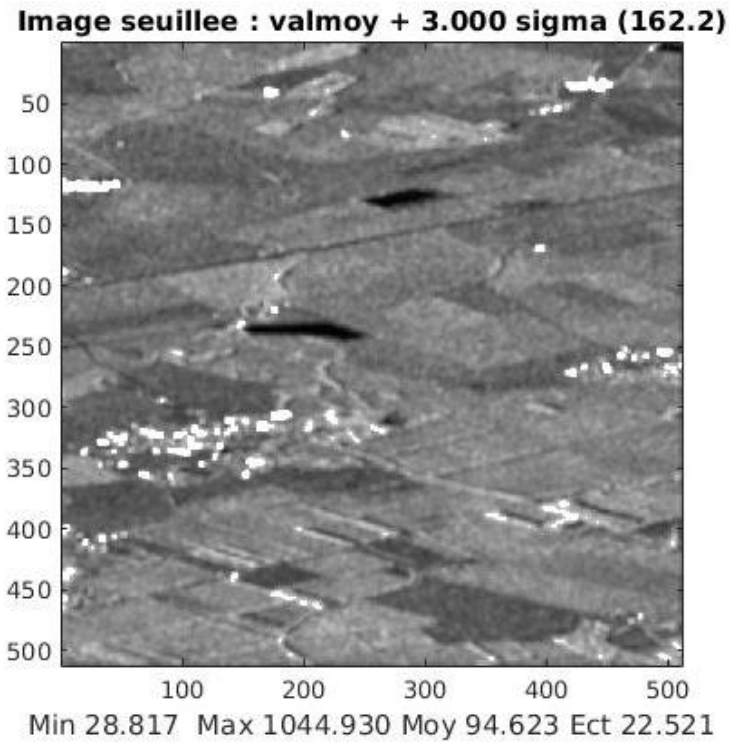
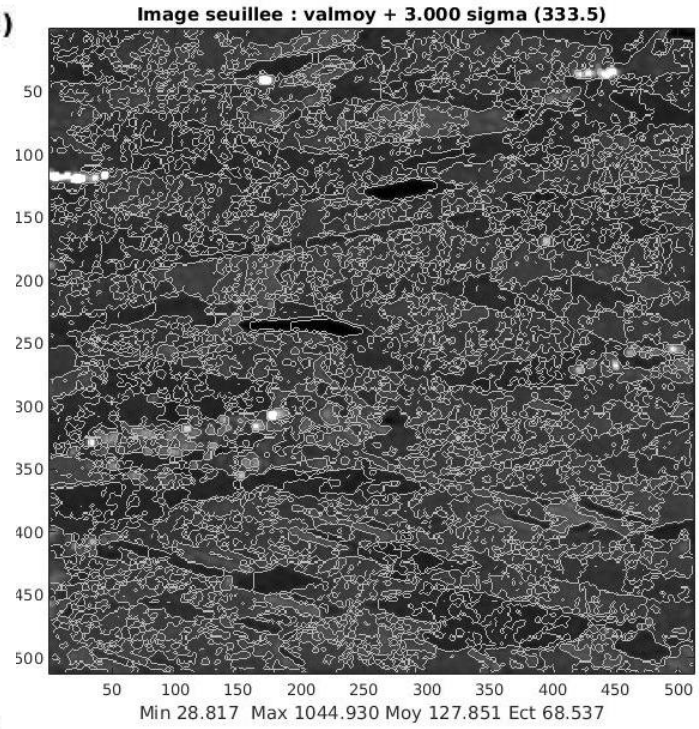
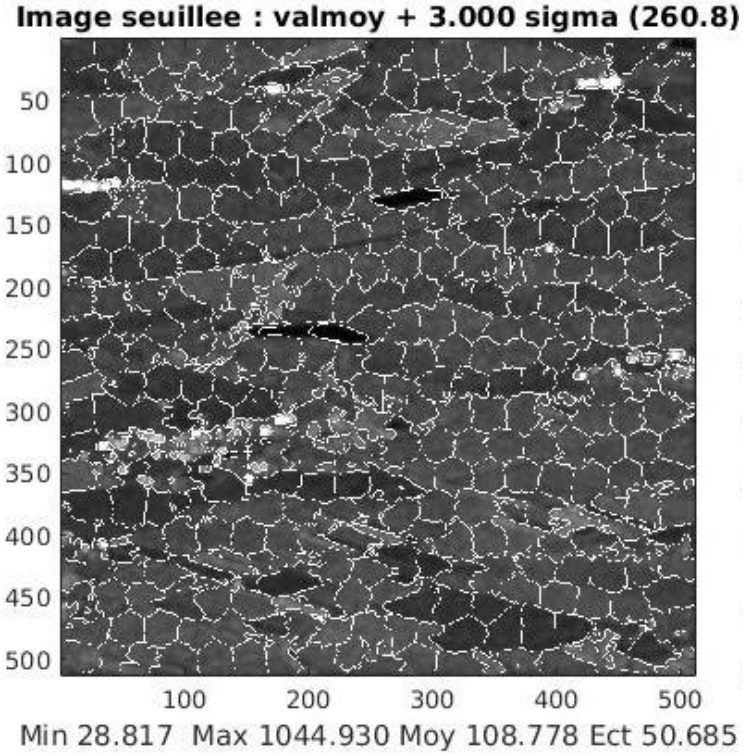


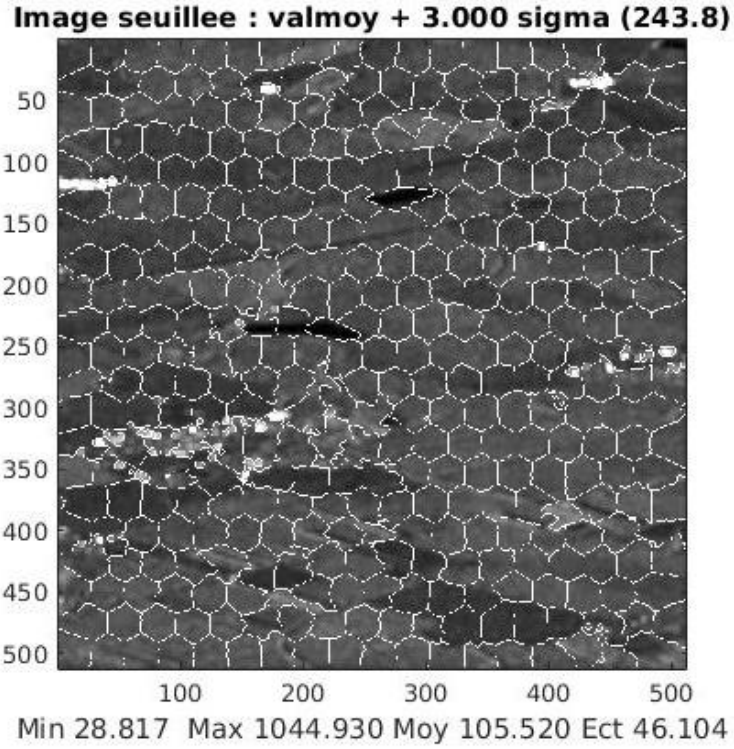
Image moyennée 5x5



m=5, 400 superpixels



m=20, 400 superpixels



m=40, 400 superpixels

VI. Conclusions

Comme on peut le voir avec ces exemples, la segmentation SLIC fonctionne bien pour les images SAR, mais si on ne filtre pas l'image avant les résultats sont pas du tout bons. Si l'image est trop bruitée, les superpixels obtenus sont trop petits et de tailles et formes irrégulières. Pour éviter cela nous pouvons aussi augmenter le paramètre m, au risque de perdre une bonne adhésion aux bords de l'image comme on le voit dans la dernière image (m=40).

Pour avoir une bonne segmentation d'une image SAR multi-temporelle il faut donc filtrer l'image pour réduire le bruit, puis choisir un bon compromis entre régularité des superpixels et adhésion aux contours en choisissant le paramètre m.

VII. Annexe

K-means :

```
function [ newImage ] = k_means( I, K, MaxIter )

I=abs(I);
I=double(I);
im=I(:);
m=floor(max(im))+1;

[hh,bins]=hist(im,m);
ind=find(hh);  % niveaux de gris non nuls qui apparaissent dans l'image

% initialisation de la valeur de chaque classe
classes=zeros(1,K);
a=0;
total=length(im);
j=1;
for i=1:length(ind)
    a=a+hh(ind(i));
    if (a/total>j/(K+1))
        classes(j)=ind(i);
        j=j+1;
    end
    if (j==K+1)
        break
    end
end

l=length(hh);
hc=zeros(1,l);

j=0;
while(j<MaxIter)
  for i=1:length(ind)
      c=abs(ind(i)-classes);
      cc=find(c==min(c));
      hc(ind(i))=cc(1); % hc contient la classe la plus proche de chaque niveau de gris
  end
  %calcul de nouvelles classes
  for i=1:K,
      a=find(hc==i);   % a contient les niveaux de gris qui feront partie de
la classe i
      d=classes(i);
      classes(i)=sum(a.*hh(a))/sum(hh(a)); % calcul de la nouvelle moyenne
      if (isnan(classes(i)))
        classes(i)=d;
      end
  end
  j=j+1;
end

% calcul de la nouvelle image
s=size(I);
newImage=zeros(s);
for i=1:s(1),
for j=1:s(2),
  c=abs(I(i,j)-classes);
  cc=find(c==min(c));
  newImage(i,j)=classes(cc(1));
end
end
end
```

SLIC :

```
function [ newImage ] = slic( I, K, MaxIter, m )

% k doit etre un carre parfait
sqrk=sqrt(K);

s=size(I);
S=s(1)/sqrk;
S=floor(S);

d=zeros(s);   %& distance
d(1:s(1),1:s(2))=-1;
L=zeros(s);   % labels
L(1:s(1),1:s(2))=-1;

% initaliser les clusters

clusters=zeros(K,3);
x0=floor(S/2);
y0=floor(S/2);
for i=1:sqrk
    for j=0:sqrk-1
        clusters(i+j*sqrk,1)=x0+(i-1)*S;
        clusters(i+j*sqrk,2)=y0+j*S;
        clusters(i+j*sqrk,3)=I(clusters(i+j*sqrk,1),clusters(i+j*sqrk,2));
    end
end


p=0;
while(p<MaxIter)
    for k=1:K
        xk=clusters(k,1);
        yk=clusters(k,2);

        for i=xk-S:xk+S
            for j=yk-S:yk+S
                if (i>=1 && i<=s(1) && j>=1 && j<=s(2))

                    dc=sqrt((clusters(k,3)-I(i,j))^2);
                    ds=sqrt((i-xk)^2+(j-yk)^2);
                    D=sqrt(dc^2+((ds*m)/S)^2);

                    if (D<d(i,j) || d(i,j)<0)
                        d(i,j)=D;
                        L(i,j)=k;
                    end
                end
            end
        end
    end

    %calcul de nouveaux clusters
    for i=1:K,

        x=0;
        y=0;
        color=0;
        n=0;

        for l=1:s(1)
            for t=1:s(2)
                if L(l,t)==i
                    x=x+l;
                    y=y+t;
```

```
                    color=color+I(l,t);
                    n=n+1;
                end
            end
        end
        clusters(i,1)=floor(x/n);
        clusters(i,2)=floor(y/n);
        clusters(i,3)=color/n;
    end
    p=p+1;
end

newImage=zeros(s);
for i=1:s(1),
for j=1:s(2),
  %newImage(i,j)=clusters(L(i,j),3);
  newImage(i,j)=L(i,j);
end
end
end
```