# Projet de Traitement du Signal
# Segmentation d'image SAR
# Rapport

Philippe Tran Ba & Élie Bouttier

10 juin 2012

**Résumé**

This report deals with a way to detect the edge using the normalized Ratio Of Exponentially Weighted Average (ROEWA) on opposite sides of the central pixel. It thus produces a map of edge with their intensity. In order to improve the result, it can be use in both direction horizontal and vertical, and the magnitude of the two components yields an edge strength map.

This edge detector can cope with the presence of speckle, which can be modeled as a strong multiplicative noise. So, It should be used for instance on SAR image. This report describe step by step the processus used in order to simulate a segmentation with MatLab.

**Résumé**

Ce rapport traite d'une méthode de détection de rupture utilisant le rapport des moyennes pondérées exponentiellement (ROEWA) normalisé sur chaque côté d'un pixel central. Elle permet ainsi d'obtenir une carte des ruptures en intensité. Pour améliorer son efficacité, le processus peut-être appliqué suivant plusieurs directions, par exemple horizontalement et verticalement, les différentes composantes étant ensuite moyennées quadratiquement.

Ce détecteur de rupture est particulièrement adapté pour traiter les signaux avec présence de bruit speckle multiplicatif, tel que les images radars à synthèse d'ouverture (Synthetic Aperture Radar — SAR). Ce rapport décrit étape par étape la simulation d'une segmentation sous MaltLab.

## Table des matières

# 1 Introduction

## 1.1 Préface

En analyse d'images, la segmentation est une étape essentielle, préliminaire a des traitements de haut niveau tels que la classification, la détection ou l'extraction d'objets. Elle consiste à décomposer une image en régions homogènes. Les deux principales approches sont l'approche région et l'approche contour. L'approche région cherche à regrouper les pixels présentant des propriétés communes tandis que l'approche contour vise à détecter les transitions entre régions. Des détecteurs efficaces ont été développés dans le cadre de l'imagerie optique, mais s'avèrent inadaptés aux images SAR de par la présence d'un bruit speckle multiplicatif.

## 1.2 Objectif du projet

L'objectif de ce projet est d'effectuer la segmentation d'une image radar a synthèse d'ouverture (image SAR) à l'aide d'une méthode originale de détection de ruptures appliquée successivement sur les lignes et colonnes de l'image. La méthode est issue d'une publication intitulée "An Optimal Multiedge Detector for SAR Image Segmentation" publiée en mai 1998 dans la revue IEEE Transactions on Geoscience and Remote Sensing.

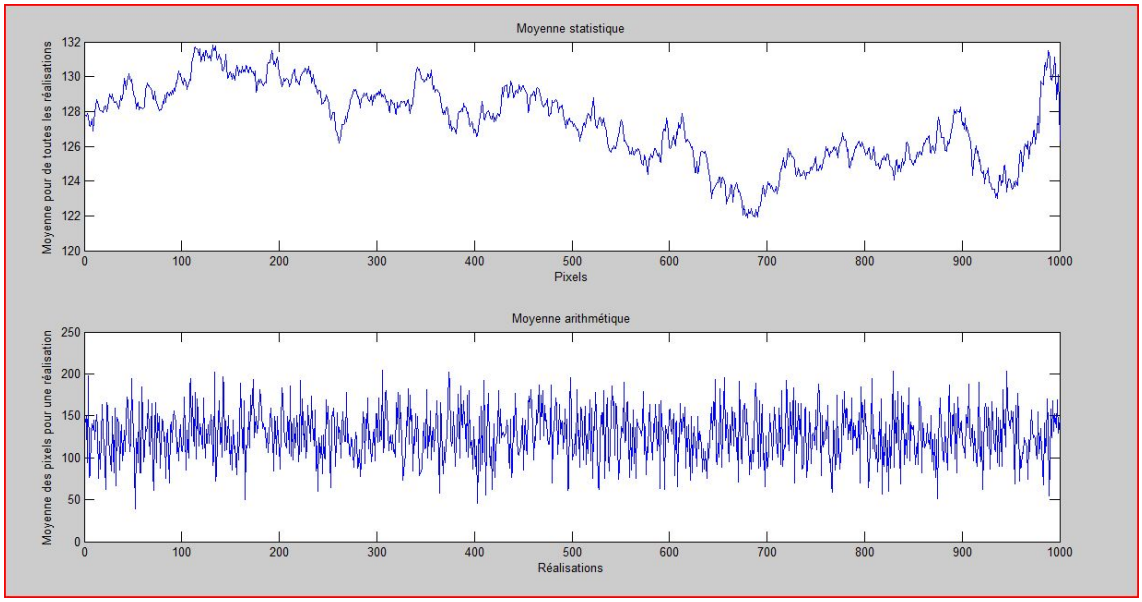
# 2 Génération d'une ligne d'image SAR

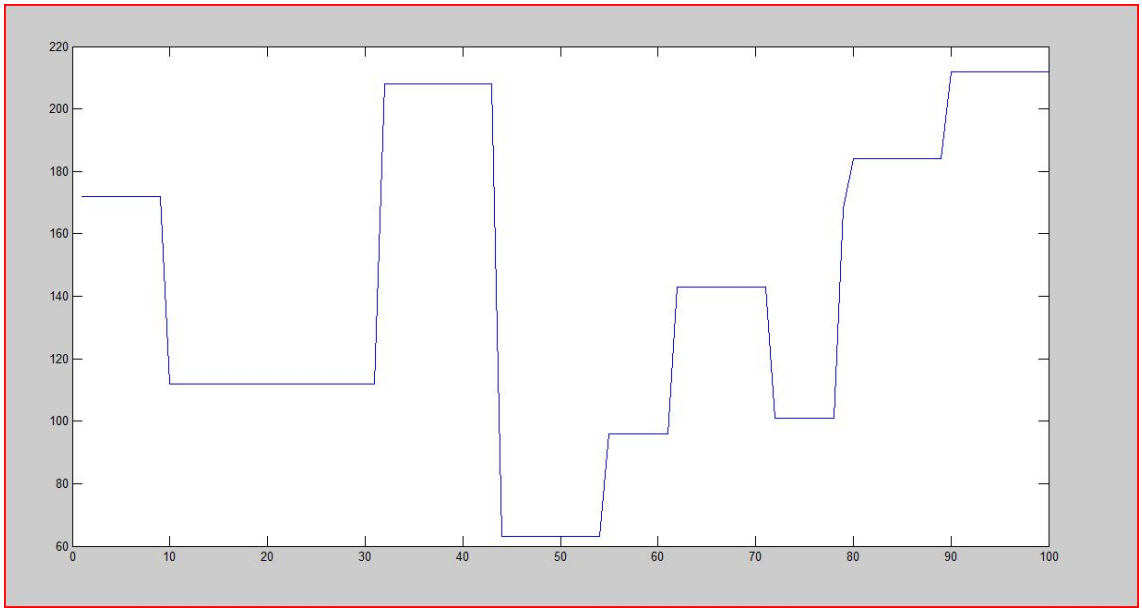
Cette partie consiste a générer des lignes d'image radar conformément au modèle proposée par l'article. La méthode de segmentation choisie sera d'abord testée sur ces lignes avant d'être appliquée a des images entières.

## 2.1 Ligne d'image non bruitée R(x)

Une ligne d'image apparaît comme une juxtaposition de segments de réflectivité constante. Elle est correctement modélisée comme un processus constant par morceaux dont les sauts d'intensité obéissent a un processus de Poisson de paramètre $\lambda$. La largeur des segments obéit alors à une loi exponentielle de paramètre $\lambda$, $\frac{1}{\lambda}$ représentant le nombre moyen de pixels séparant deux sauts d'intensité. On utilise donc le code MatLab suivant afin de générer une ligne d'image non bruitée avec 255.

```
mu = 10; %parametre pour la loi exponentielle
N = 500; %nombre de pixel
k = 1; %compteur a ne pas toucher
Suite_symbole = zeros(1,N); %Preallocation de la memoire

while(k<N)
    i = floor(rand*256);   %Une intensite entre 0 et 255
    for j = 1:ceil(exprnd(mu))
        Suite_symbole(k) = i; %propager i sur j pixel
        k = k+1;
    end
end

R = Suite_symbole(1:500);   %On ampute le signal des valeurs en trop

plot(R)
```

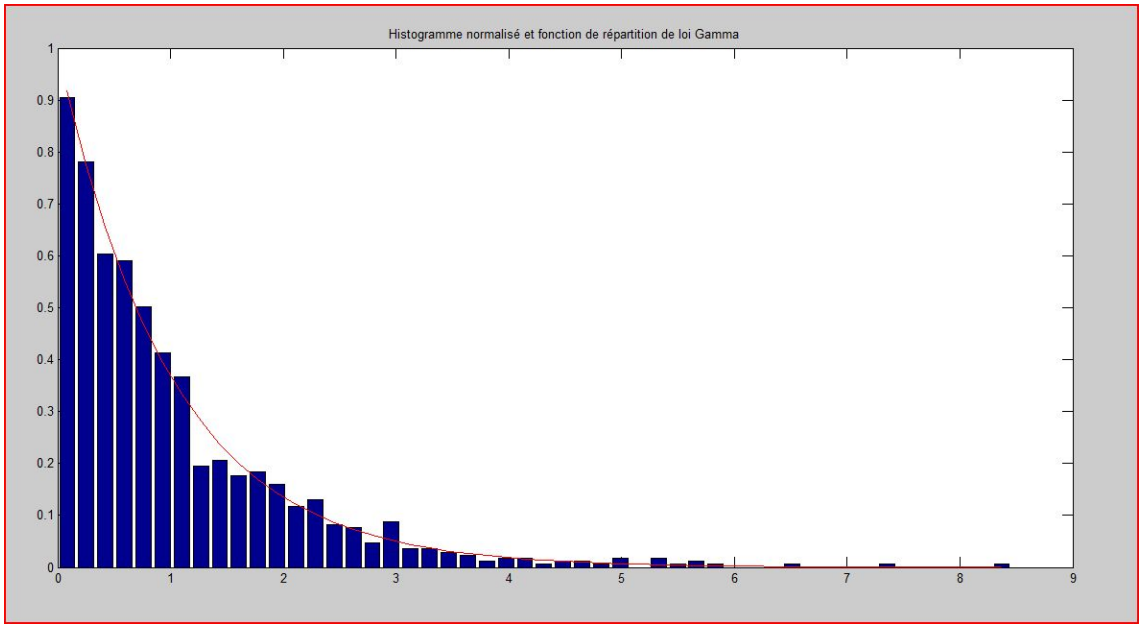





## 2.2 Bruit Speckle Multiplicatif n(x)

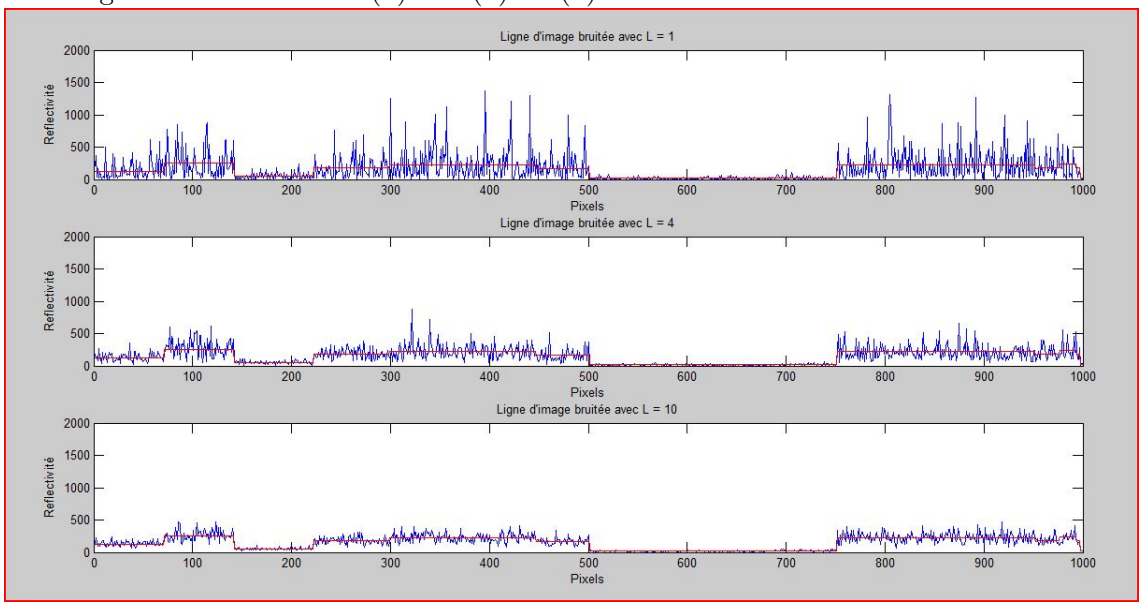
Pour modéliser le bruit speckle, on utilisera une suite de variables aléatoires indépendantes suivant une loi Gamma de moyenne $\mu_n = 1$ et de variance $\sigma_n^2 = 1/\mathrm{L}$, où L correspond au nombre de vue moyennée.

```
L = 1; %nombre de vues moyennees
N = 500; %nb de valeur
Nc = 80; %nombre de classe

%Generer l'histogramme normalise
bruit_mult = gamrnd(L, 1/L, 1, N);
[hist abs] = hist(bruit_mult,Nc);
bruit_th = gampdf(abs, L, 1/L);
Nc = 50;
dx = (max(bruit_mult)-min(bruit_mult))/Nc;
%Affichage de l'histogramme normalise
figure('name', 'Bruit multiplicatif n(x)')
bar(abs, hist/(dx*length(bruit_mult)));
title('Histogramme normalise et fonction de repartition de loi Gamma')
hold on
%Affichage de la fonction de repartition theorique
plot(abs, bruit_th, 'r');

n = bruit_mult;
```




## 2.3 Ligne d'image bruitée I(x)

Pour obtenir une ligne d'image altérée par une bruit multiplicatif, il suffit de multiplier la ligne d'image et le bruit. Soit : I(x) = R(x) * n(x).



# 3 Analyse Spectrale d'une ligne d'image SAR

Dans cette partie, le signal synthétique est étudié à l'aide des outils classiques d'analyse spectrale (corrélogramme, périodogramme).

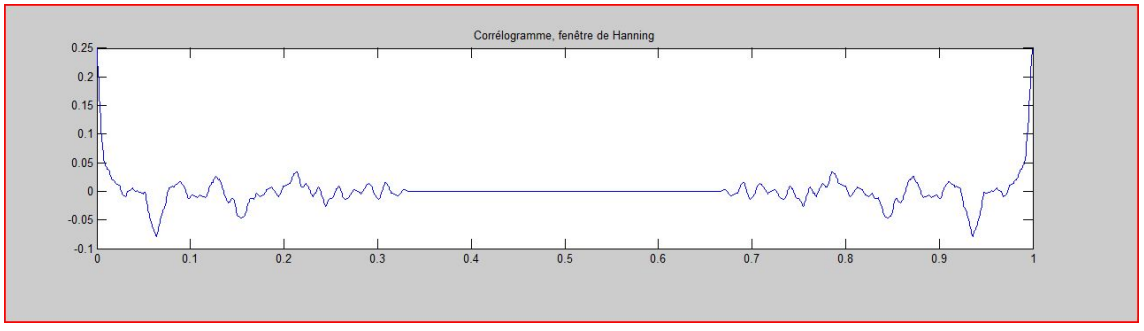
## 3.1 Périodogramme

```
function [ perio ] = periodogramme( lambda, largeur, profondeur, fenetre, bruit)
    % Generation du periodogramme

    % Generation d'une ligne
    if nargin < 5
        ligne = genligne(lambda, largeur, profondeur);
    else
        ligne = genlignebruite(lambda, largeur, profondeur, bruit);
    end

    % On rend la moyenne nulle
    % (Pour supprimer le terme constant dans le periodogramme)
    ligne = ligne - mean(ligne);

    % Fenetrage
    ligne = ligne .* fenetre;

    % Generation du periodogramme
    perio = (abs(fft(ligne)).^2)./largeur;

end
```

## 3.2 Périodogramme cumulé

```
function [ p ] = periocumule( lambda, largeur, profondeur, fenetre, cumule, bruit )
    % Creer un periodogramme cumule

    p = zeros(1, largeur); % Preallocation de la memoire
    for i = 1:cumule;
        % Ajout d'un nouveau periodogramme
        if nargin < 6
            p = p + periodogramme(lambda, largeur, profondeur, fenetre);
        else
            p = p + periodogramme(lambda, largeur, profondeur, fenetre, bruit);
        end
    end

    % Moyennage
    p = p ./ cumule;

end
```

## 3.3 Corrélogramme



```
function [ corr_sym ] = correlogramme( lambda, largeur, profondeur, fenetre, zlargeur )

    ligne = genligne(lambda, largeur, profondeur);

    ligne = ligne .* fenetre;

    corr = xcov(ligne, 'biased');

    part1 = corr(1, largeur:end);
    part2 = zeros(1, zlargeur);
    part3 = 0;
    part4 = part2;
    part5 = corr(1, end-1:-1:largeur);

    corr_sym = [part1 part2 part3 part4 part5];

end
```

# 4 Détection de Ruptures sur une ligne d'image SAR

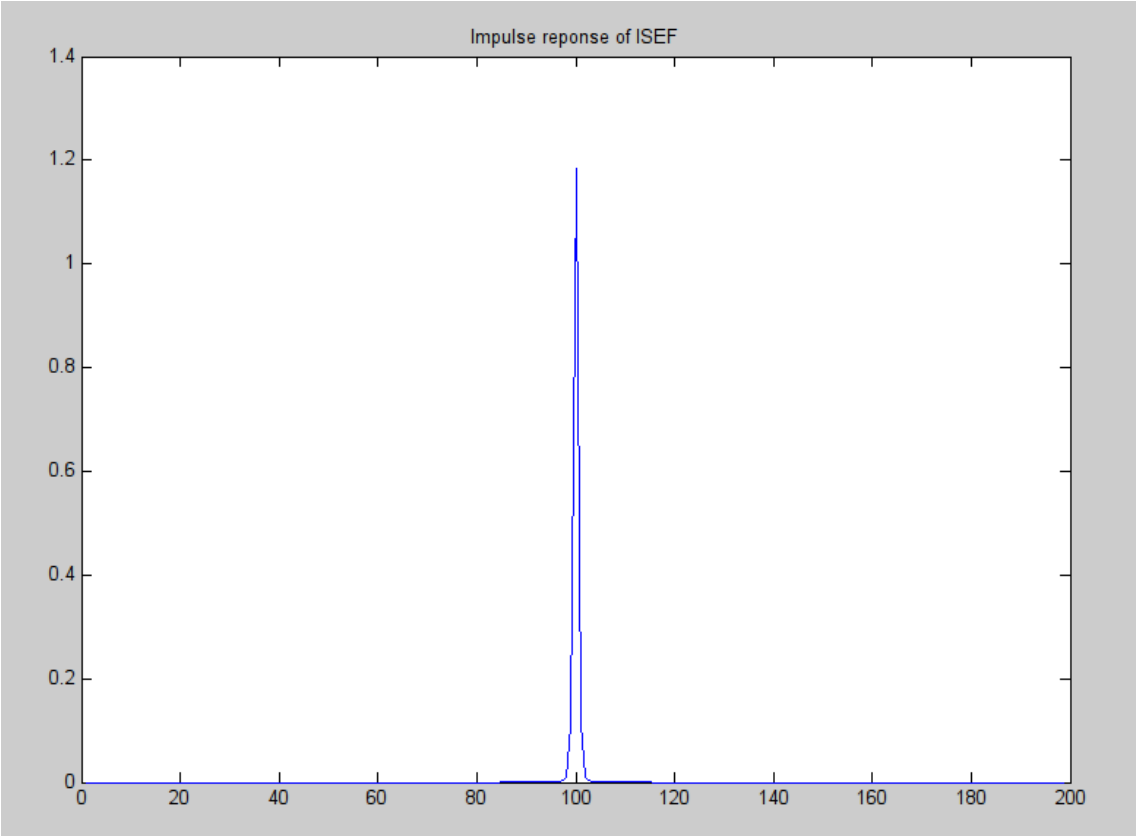
Le détecteur ROEWA est appliqué au signal simulé. Il est basé sur des contrastes locaux de niveau radiométrique moyen.

La détection s'effectue au moyen d'une fenêtre d'analyse glissante. Une forte différence de réflectivité moyenne de part et d'autre d'un pixel permet de repérer un contour. Cette méthode, bien adaptée aux images optiques, est mise en défaut pour l'imagerie radar. La présence d'un bruit multiplicatif augmente en effet le taux de fausses détections dans les régions de forte réflectivité. Pour pallier cette limitation, l'article propose un détecteur basé non plus sur des différences, mais sur des rapports de réflectivité moyenne. En outre, les moyennes arithmétiques sont remplacées par des moyennes pondérées exponentiellement pour traiter les cas de contours multiples (présence de plusieurs contours dans la fenêtre d'analyse). Les plus proches voisins du pixel central sont ainsi favorisés aux dépens de pixels plus éloignés pouvant correspondre a un nouveau contour.

Le code MatLab ci dessous nous permet de synthétiser un filtre ISEF et de l'utiliser sur une ligne bruitée.

```
%
% Parametres
profondeur = 32 % Precision de la quantification
N = 2000 % Taille de notre ligne en pixel
mu = 150 % Nombre moyen de pixel entre deux ruptures d'intensite
lambda = 1/mu
L = 500 % Parametre du bruit

% Generation d'une ligne de pixels
ligne = genligne(lambda, N, profondeur);
subplot(3, 1, 1);
plot(ligne);
title('Ligne non bruite');

% Generation du bruit de scintillement
bruit = gamrnd(L, 1/L, 1, N);
subplot(3, 1, 2);
plot(bruit)
title('Bruit de scintillement');

% Multiplication de notre ligne par le bruit
ligne_bruite = ligne .* bruit;
subplot(3, 1, 3);
plot(ligne_bruite);
title('Ligne bruite');
figure;


%
% Parametres
muI = mean(ligne);
sigmaI = sqrt(var(ligne_bruite));

muR = muI;
sigmaR = (L*sigmaI^2 - muI^2) / (L + 1);

alpha = sqrt(((2*L*lambda) / (1 + (muR/sigmaR)^2)) + lambda^2);
C = alpha/2;

order = 199

% Generation de la reponse impulsionnelle
ISEF = C * exp(-alpha*abs([-(order-1)/2:1:order/2]));
plot(ISEF);
title('Impulse reponse of ISEF');

% Filtrage du signal. Celui-ci est alonge de la longueur du retard afin
% de pouvoir effectuer le filtrage jusqu'au bout.
ligne_debruite = filter(ISEF, 1, [ligne_bruite zeros(1, ceil(order/2))]);
```

```
ligne_debruite = ligne_debruite(1, ceil(order/2):end); % Suppression du retard
figure
plot(ligne_bruite, 'b');
hold on
plot(ligne_debruite, 'r');
title('Comparaison signal bruite et signal debruite');
legend('Signal bruite', 'Signal debruite');
```




Grâce au filtre ISEF, le bruit est fortement atténué et le signale est bien plus lisible. Nous allons donc pouvoir procéder à la détection de rupture par l'opérateur ROEWA noté $r_{max}$. Nous utilisons donc la suite du programme suivante afin de calculer le tableau des ROEWA :

```
%
% Separation des parties droite et gauche du filtre ISEF
ISEF1 = [ISEF(1, 1:ceil(order/2)) zeros(1, ceil(order/2))];
ISEF2 = [zeros(1, ceil(order/2)) ISEF(1, ceil(order/2):end)];

% Filtrage du signal par les deux filtres
mu1 = abs(filter(ISEF1, 1, [ligne_debruite zeros(1, floor(order/2))]));
mu1 = mu1(1, floor(order/2):end); % Suppression du retard
mu2 = abs(filter(ISEF2, 1, [ligne_debruite zeros(1, floor(order/2))]));
mu2 = mu2(1, floor(order/2):end); % Suppression du retard

figure;
subplot(5, 1, 1);
plot(mu1);
title('mu1');
subplot(5, 1, 2);
plot(mu2);
title('mu2');
subplot(5, 1, 3);
plot(mu1./mu2);
title('mu1/mu2');
subplot(5, 1, 4);
plot(mu2./mu1);
title('mu2/mu1');
subplot(5, 1, 5);
% Calcul de Rmax
rmax = max(mu1./mu2, mu2./mu1);
```

```
plot(rmax);
title('Rmax');

% Observation de Rmax supperpose avec le signal bruite
figure;
plot(ligne_bruite, 'b');
hold on;
plot(rmax, 'r', 'LineWidth', 1.5);
legend('Signal bruite', 'Rmax');
hold off
```

Enfin, pour terminer, nous déterminons les ruptures à partir d'un seuil de ROEWA. Nous affichons suite au programme MatLab une figure présentant ROEWA, les ruptures détectées ainsi que le signal non bruité.

```
%
seuil = 1.2 % Seuil de detection des transitions

p = 0;
trans = [];

figure
plot(ligne, 'b', 'LineWidth', 1); % Signal non bruite
hold on
plot(rmax-2, 'r'); % Rmax (-2 pour eviter les superpositions)

for i = 2:N-1;
    % Detection des maximums locaux
    if rmax(i-1)<rmax(i) && rmax(i)>rmax(i+1);
        % Verification que le maximum depasse le seuil
        if rmax(i) > seuil;
            p = p+1;
            trans(p) = i-1;
            x = [trans(p);trans(p)];
            y = [min(ligne(i-2:i+2)), max(ligne(i-2:i+2))];
            line(x, y, 'color','g', 'LineWidth', 2)
        end
    end
end
legend('Signal (avant bruitage)', 'Rmax', 'Transitions');

trans
```

# 5 Détection de Ruptures sur une image TEST

La segmentation d'une image est réalisée en plusieurs étapes. Les ruptures sont détectés successivement en ligne et en colonne de façon a créer une carte des contours délimitant les régions de radiométrie différente.

La détection de contours sur une image est réalisée en deux temps. Le détecteur ROEWA est d'abord appliqué successivement a chaque ligne de l'image, préalablement lissée dans la direction opposée, pour obtenir la carte horizontale des contours $r_X(x, y)$. En opérant de façon similaire dans la direction verticale, la carte verticale des contours est construite. Ces deux composantes peuvent alors être combinées pour former la carte des contours en deux dimensions : $r_{2D}(x, y) = \sqrt{r^2{}_X + r^2{}_Y}$

# 6 Conclusion