TP Traitement du signal
Application de méthodes de recalage d'images

31 août 2020

Objectifs

L'objectif final est de réaliser un algorithme de recalage d'image. Suivant ce schéma fonctionnel :

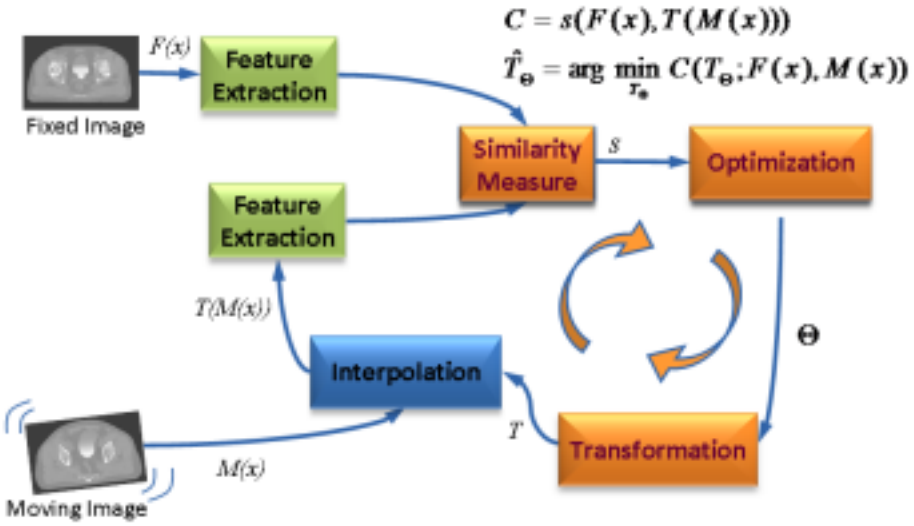


FIGURE 1 – Schéma fonctionnel de l'algorithme de recalage

F(x) est un élément d'un ensemble d'images et T(M(x)) est une transformation par rotation de cet élément, l'objectif est d'estimer la valeur de rotation θ de cette transformation. C'est un problème d'optimisation dont la fonction de coût S est générée empiriquement par la fonction "Similarity Measure" qui conceptuellement correspond à une mesure de "distance" entre deux éléments dans l'ensemble des caractéristiques C. Dans ce TP l'ensemble des caractéristiques correspondent à l'image elle-même. Cette fonction de coût est donc généré par évaluations successives entre l'élément F(x) et l'élément T(M(x)) transformé par θ, la valeur rechercher correspondra à la valeur de θ qui minimisera la mesure.

Dans ce TP on essayera de comparer plusieurs fonctions de similarité avec des stratégies d'optimisation différentes.

I est l'ensemble des images : $I \subset M_{157,103}(\{0,..,255\})$

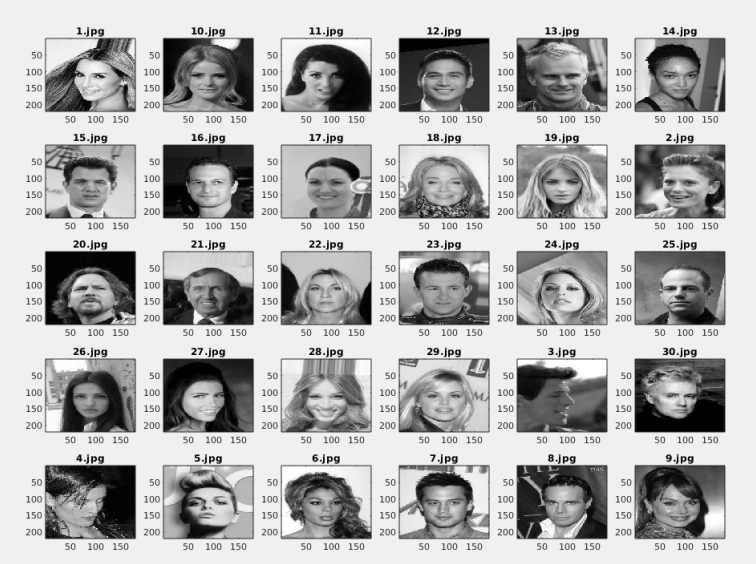


FIGURE 2 – Partie des images non transformées dans l'ensemble I

Le reste des images peut êtres généré par le programme "MyRotate.m".

Mesure de similarité

Une mesure de similarité S est une fonction $C \times C \to \mathbb{R}$

Afin d'implémenter un algorithme d'optimisation de façon à être invariant aux choix de ces fonctions, elles possèderons ces propriétés.

$\forall x, y \in I, S(x,y) = S(y,x)$ Symétrie
$\forall x, y \in I, S(x,x) \leq S(y,x)$ Minimalité
$\forall x, y \in I, S(x,y) \geq 0$ Positivité

Somme de différences au carré (SSD)

$SSD(I_1, I_2) = \frac{(I_1 - I_2)^2}{n*m}$ avec $I_1, I_2 \in I$ et m, n la dimension des images.

Listing 1 – Fonction de mesure de similarité SSD

```
classdef SSD < Measure

    methods
        function [obj] = measure(obj, image1, image2)
            [m,n] = size(image1);
            obj.result = sqrt(sum(sum(((image1-image2).^2))/(m*n)));
        end
    end
end
```

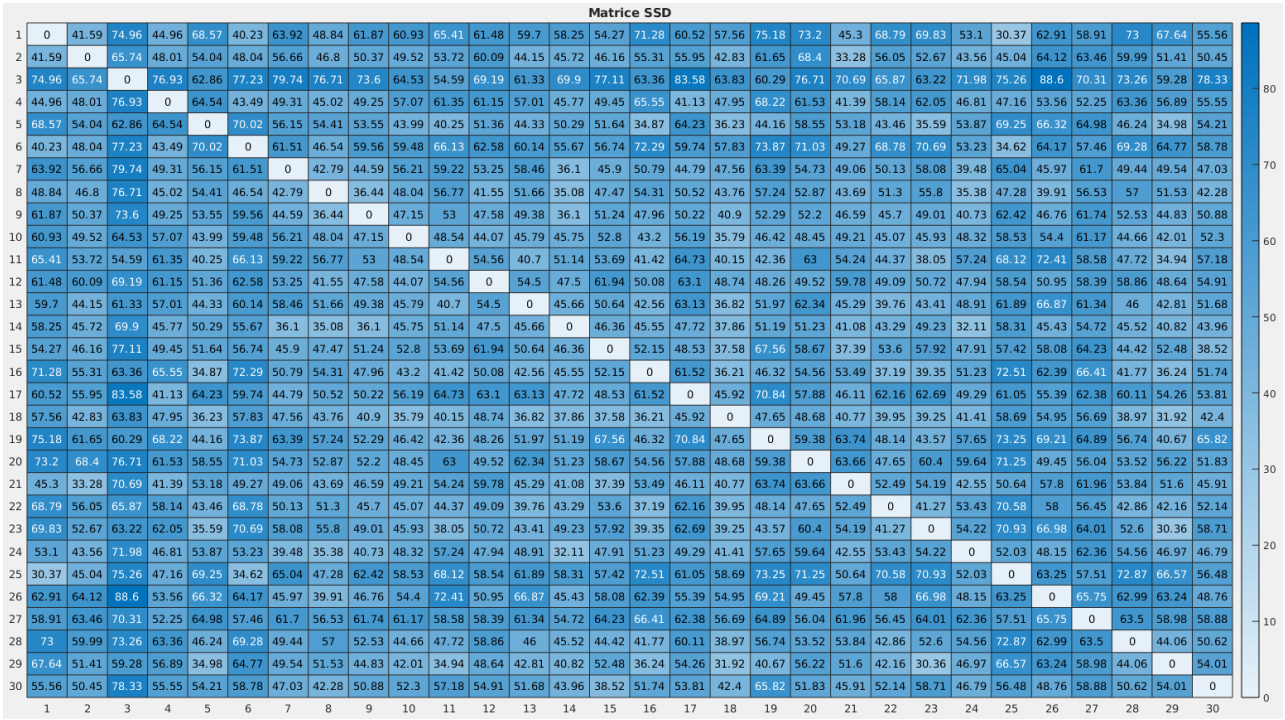


FIGURE 3 – Mesure SSD de l'ensemble des couples d'images non transformées

Entropie conjointe

$I_1, I_2 \in I$ sont vues comme des variables aléatoires P_{i_1,i_2} corresponds à l'histogramme conjoint.

$H(I_1, I_2) = -\sum_{i_1,i_2} P_{i_1,i_2}.log_2(P_{i_1,i_2})$

Listing 2 – Fonction de mesure de similarité Entropie conjointe

```
classdef EntropieConjointe < Measure
    methods
        function obj = measure(obj, image1, image2)
            Pxy = EntropieConjointe.PXY(image1, image2);
            U = Pxy.*log2(Pxy);
            obj.result = -sum(U(~isnan(U)));
        end
    end
    methods(Static)
        function [Pxy] = PXY(image1, image2)
            [n,m] = size(image1);
            Pxy = zeros(256, 256);
            for x = 1:n
                for y = 1:m
                    idx1 = image1(x,y)+1;
                    idx2 = image2(x,y)+1;
                    Pxy(idx1, idx2) = Pxy(idx1, idx2)+1;
                end
            end
            Pxy = Pxy/(m*n);
        end
    end
end
```

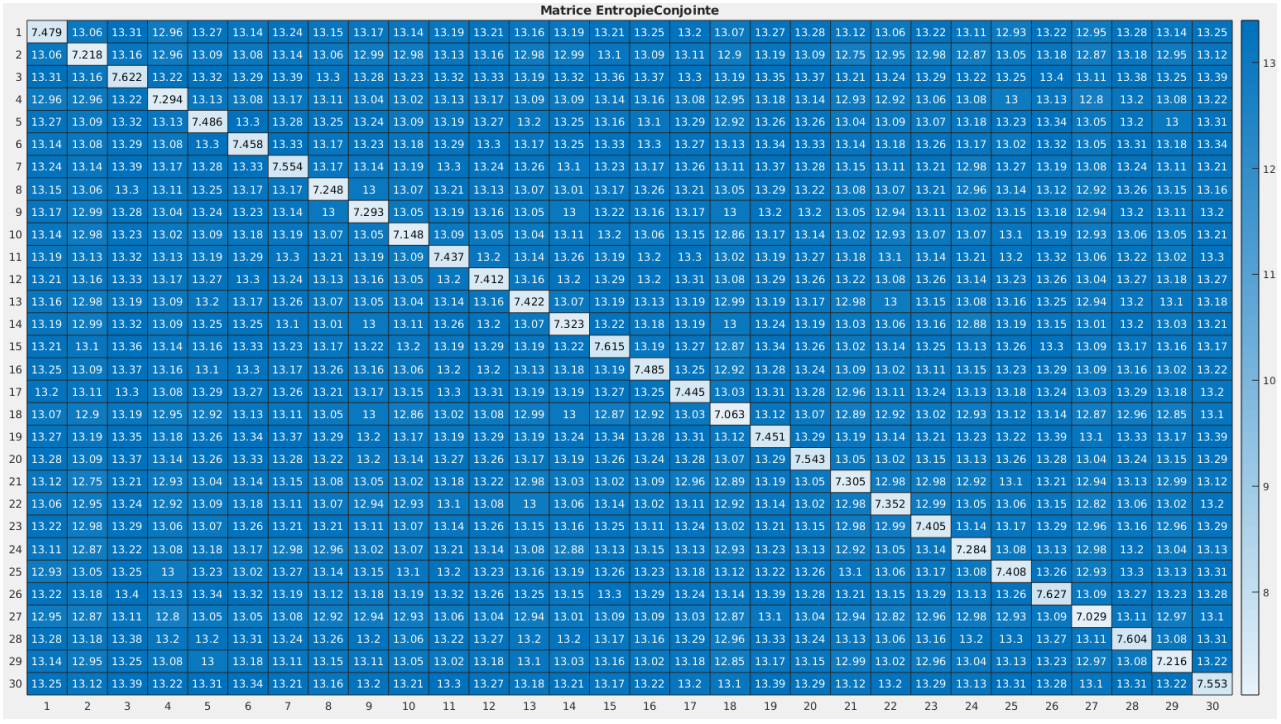


FIGURE 4 – Mesure de l'entropie conjointe de l'ensemble des couples d'images non transformées

Information mutuelle

Mesure complémentaire à la précédente, utile si la fonction de coût est à maximisé. N'est pas utilisé par la suite.

$$\forall I_1, I_2 \in I, H(I_1, I_2) \leq H(I_1) + H(I_2) \text{ avec } H(I_e) = -\sum_{i_e} P_{i_e}.log_2(P_{i_e})$$

Listing 3 – Fonction de mesure de similarité Information mutuelle

```
classdef InformationMutuelle < Measure
    methods
        function obj = measure(obj, image1, image2)
            obj.result = InformationMutuelle.Entropie(image1)...
            +InformationMutuelle.Entropie(image2)...
            - EntropieConjointe().measure(image1, image2).result;
        end
    end
    methods(Static)
        function result = Entropie(image)
            P = InformationMutuelle.PX(image);
            U = P(1, :) .*log2(P(1, :));
            result = -sum(U(~isnan(U)));
        end

        function [P] = PX(image)
            [n,m] = size(image);
            P = zeros(1, 256);
            for i = 0:255
              P(1,i+1) = sum(sum(image == i))./(n*m);
            end
        end
    end
end
```

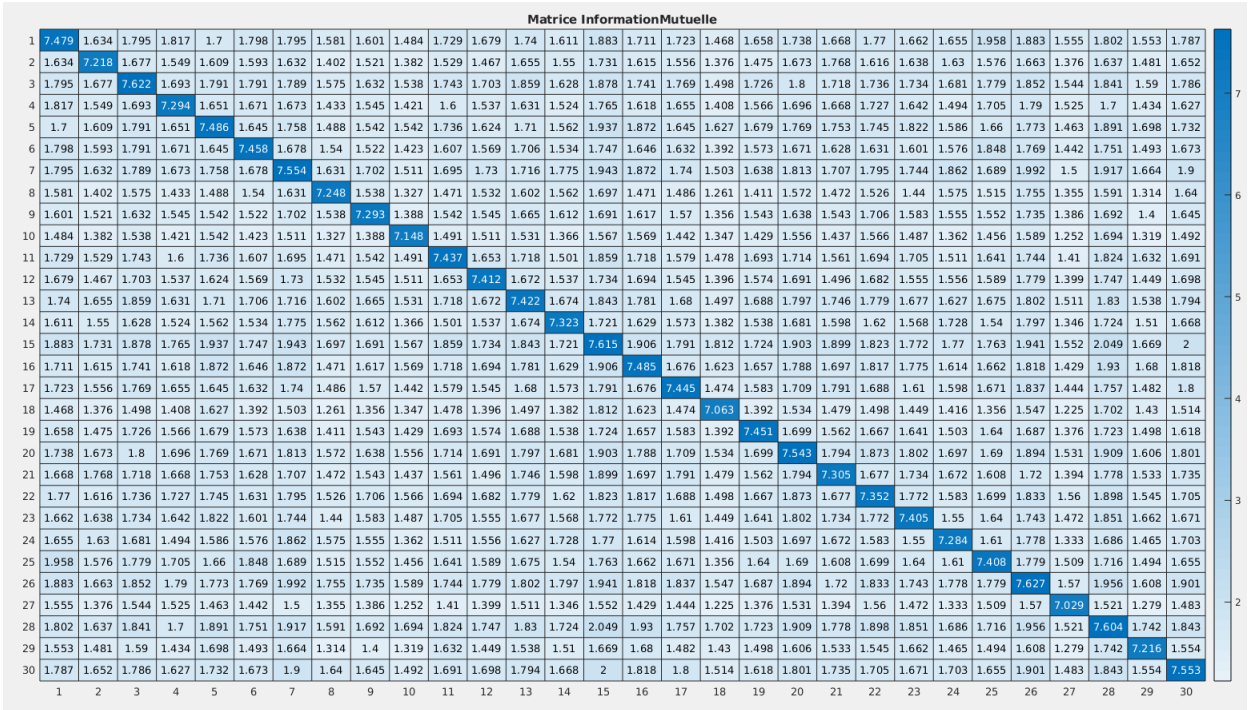


FIGURE 5 – Mesure de l'information mutuelle de l'ensemble des couples d'images non transformées

Corrélation croisée modifié

$$CC = \frac{\sum_{x,y}(I_1(x,y)-\mu_{I_1})(I_2(x,y)-\mu_{I_2})}{\sqrt{\sum_{x,y}(I_1(x,y)-\mu_{I_1})^2 . \sum_{x,y}(I_2(x,y)-\mu_{I_2})^2}}$$

Pour avoir les bonnes propriétées $CCm = (1 - CC)^2$

Listing 4 – Fonction de mesure de similarité Corrélation croisée modifié

```
classdef CrossCorrelation < Measure
    methods
        function obj = measure(obj, image1, image2)
            mu1 = mean(mean(image1));
            mu2 = mean(mean(image1));
            obj.result = (1-sum(sum((image1-mu1).*(image2-mu2)))...
            /(sqrt(sum(sum((image1-mu1).^2))*sum(sum((image2-mu2).^2)))))^2;
        end
    end

end
```

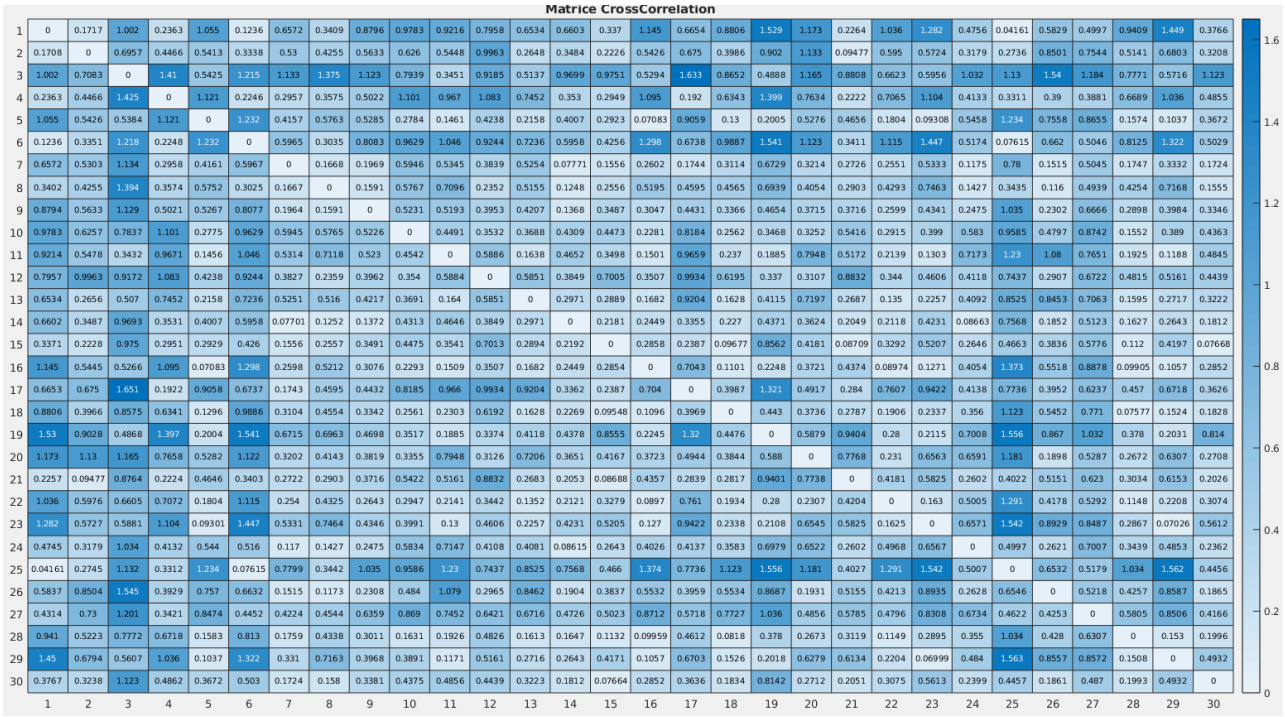


FIGURE 6 – Mesure de la corrélation croisé modifié de l'ensemble des couples d'image non transformés

Différence images

Cette mesure comptabilise le nombre de pixels différents dans l'image.

Listing 5 – Fonction de mesure de similarité Différence images

```
classdef DiffImages < Measure
    methods
        function obj = measure(obj, image1, image2)
            obj.result = sum(sum(image1 ~= image2));
        end
    end
end
```

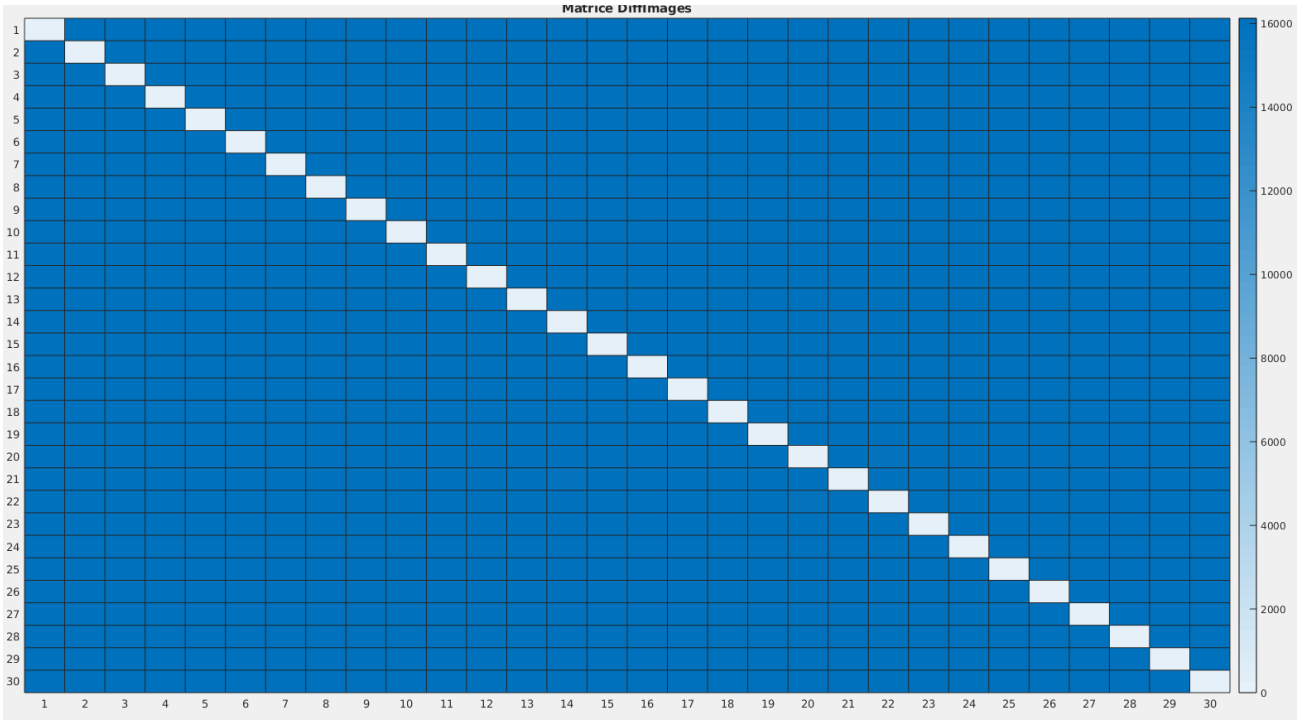


FIGURE 7 – Mesure de la différence images de l'ensemble des couples d'images non transformées

Ces figures permettent de vérifier les propriétés de positivité, symétrie et de minimalité entre images non transformées.

Optimisation

Motivation de l'utilisation de mesure de similarité

Un algorithme de recalage est avant tous un problème d'optimisation et les mesures implémentées précédemment vont permettre de le résoudre raisonnablement. En effet notre problème est d'estimer le plus précisément θ le paramètre de transformation de notre image. Si θ est un paramètre pris dans $\mathbb{R}$ alors il existera toujours une erreur, puisqu'il n'y a pas de bijection avec l'ensemble des images transformées par ce paramètre. L'image sera différente pour un $\delta\theta$ de l'ordre de 10^{-5} degrés. Cette précision n'est pas constante du fait de la troncature et pour des valeurs critiques.
Si on essayait d'estimer ces valeurs par une égalité entre l'image et l'image transformée par θ on n'aurait pas d'autre choix que de tester cette égalité avec un échantillonnage de 10^{-5} sur le domaine $[0, 360]$ ce qui correspond à 360.10^5 valeur de θ dans le pire des cas. Cette ordre de grandeur est déjà peu envisageable alors sur des problèmes de recalage plus complexes avec plusieurs paramètres à optimiser sur des domaines de recherche plus grands, la tache devient très difficile.

Les mesures de similarité vont nous permettre de nous guider dans la recherche de cette estimation. On espère d'elles qu'elle convergent vers leur valeur minimale d'une manière a adopté une bonne stratégie d'optimisation rendant la résolution du problème efficient.

Le but pour moi est donc de trouver cette égalité avec une complexité d'algorithmes la plus minimale possible, il n'est pas envisagé de modifier l'espace des caractéristiques. Le nombre d'appels de mesure de similarité sera donc la mesure de cette complexité.

Etude des fonctions de coût

Les 4 types de fonction de coût évalué sont les suivantes : (Somme de différences au carré, Entropie conjointe, Corrélation croisée modifiée, Différence images)

Pour simplifier l'étude de ces fonctions on va émettre plusieurs hypothèses ;

1. Considérer l'allure des mesures de similarité en fonction de θ invariant aux images.
2. Une stratégie d'optimisation valable pour un θ recherché sera valable pour toutes les autres valeurs. Autrement dit l'allure des convergences vers la valeur minimale ne dépend pas de la valeur recherchée.
3. L'indépendance de ces deux derniers paramètres, l'image et la rotation recherchée.

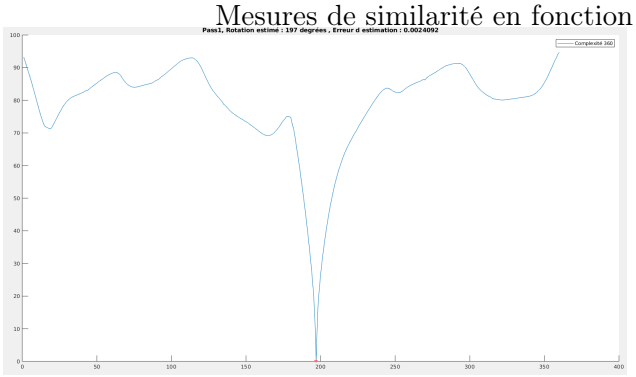
En annexe de quoi rendre raisonnables les deux premières hypothèses.

Par la suite, θ recherché et l'image sont tirées aléatoirement dans leur ensemble..

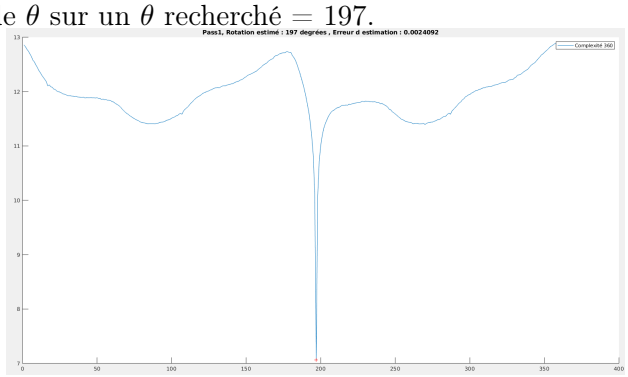
Listing 6 – Fonction de génération de l'image et de θ recherché

```
function [theta, index, F, T] = GenImages(images)
    %Problème pour rand(1) == 1 mais improbable
    index = floor(rand(1)*size(images, 3))+1;
    F = images(:,:,index);
    theta = rand(1)*360;
    T = MyRotate(theta, F, F);
end
```

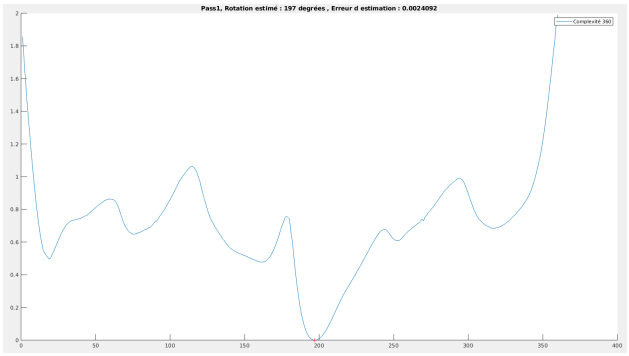
Mesures de similarité en fonction de θ sur un θ recherché = 197.



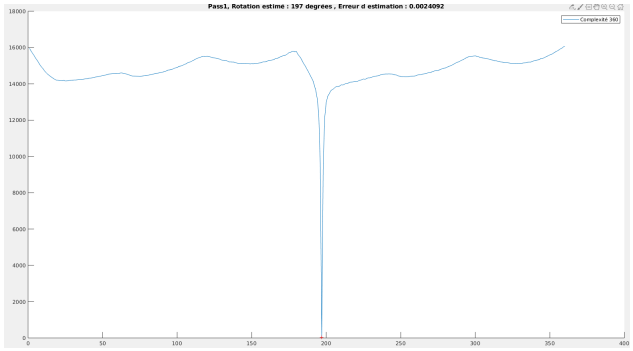
SSD



Entropie conjointe



Corrélation croisée modifié



Différence images

Cette convergence progressive vers la vraie valeur de rotation avec ces mesures, est a priori possible parce que plus l'erreur entre la rotation réelle et la rotation estimée est faible plus le nombre de pixels au centre de l'image sont à la bonne position. Cette explication est plus simplement validée en partie par la mesure Différence images qui est corrélée avec les autres mesures.(Ligne 4

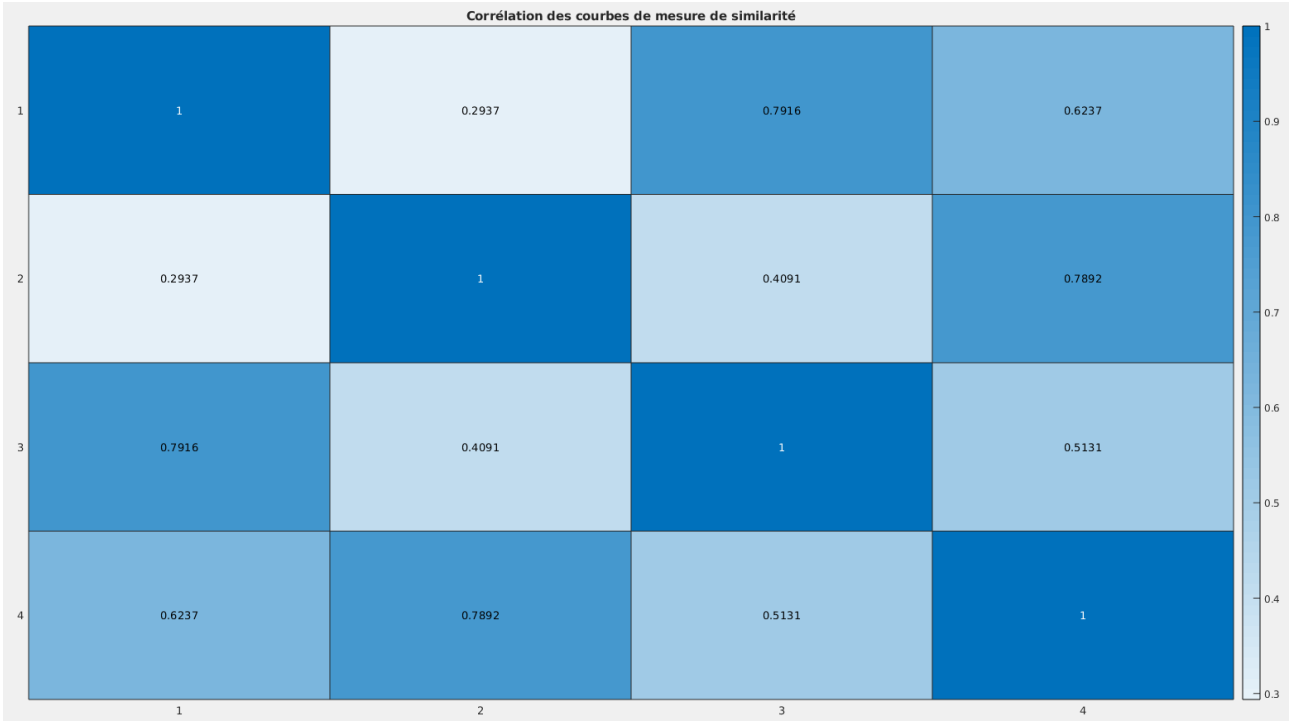
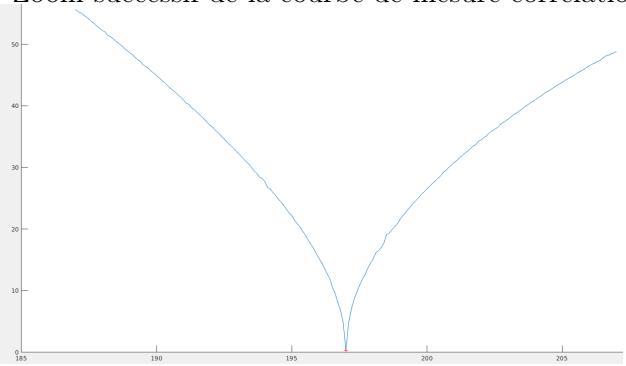
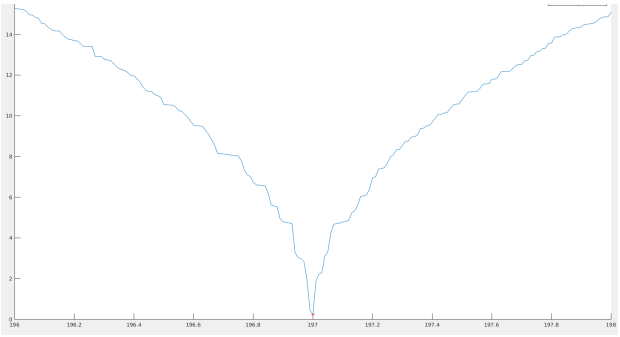


FIGURE 8 – Corrélation des courbes de mesure de similarité

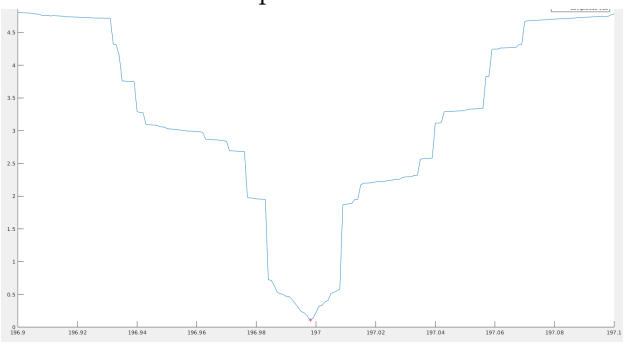
Zoom successif de la courbe de mesure corrélation croisée modifié centré sur la valeur minimale.



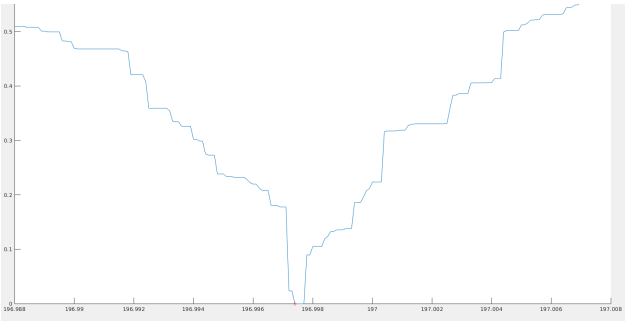
pas : 0.1



pas : 0.01



pas : 0.001



pas : 0.0001

On remarque que les courbes sont décroissantes jusqu'au point minimal et croissant au-delà. Cette observation est valable sur les autres courbes de mesure.

Stratégie d'optimisation

Avec ces observations il est très facile d'optimiser ces fonctions. Cette une optimisation d'une fonction convexe, pour ce faire on a fait le choix d'implémenter un algorithme simple en suivant ce logigramme :

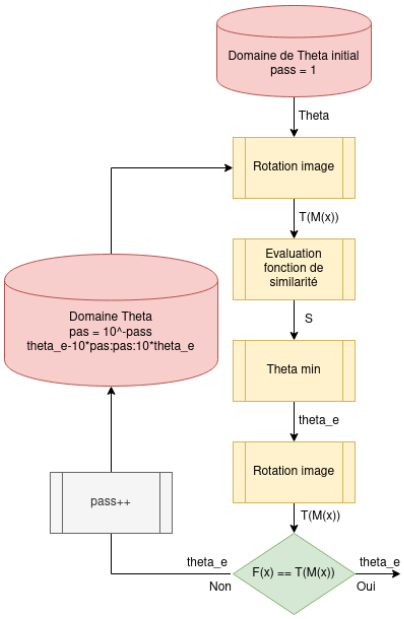


FIGURE 9 – Schéma de principe de l'algorithme d'optimisation

Il est clair qu'il faut réduire le domaine de recherche le plus tôt possible.
Pour optimiser la taille du domaine de θ initial la mesure de corrélation croisée modifiée est la meilleure, en effet avec un échantillonnage à 30 degrée on s'assure déjà de ne pas tomber dans un minimum local.

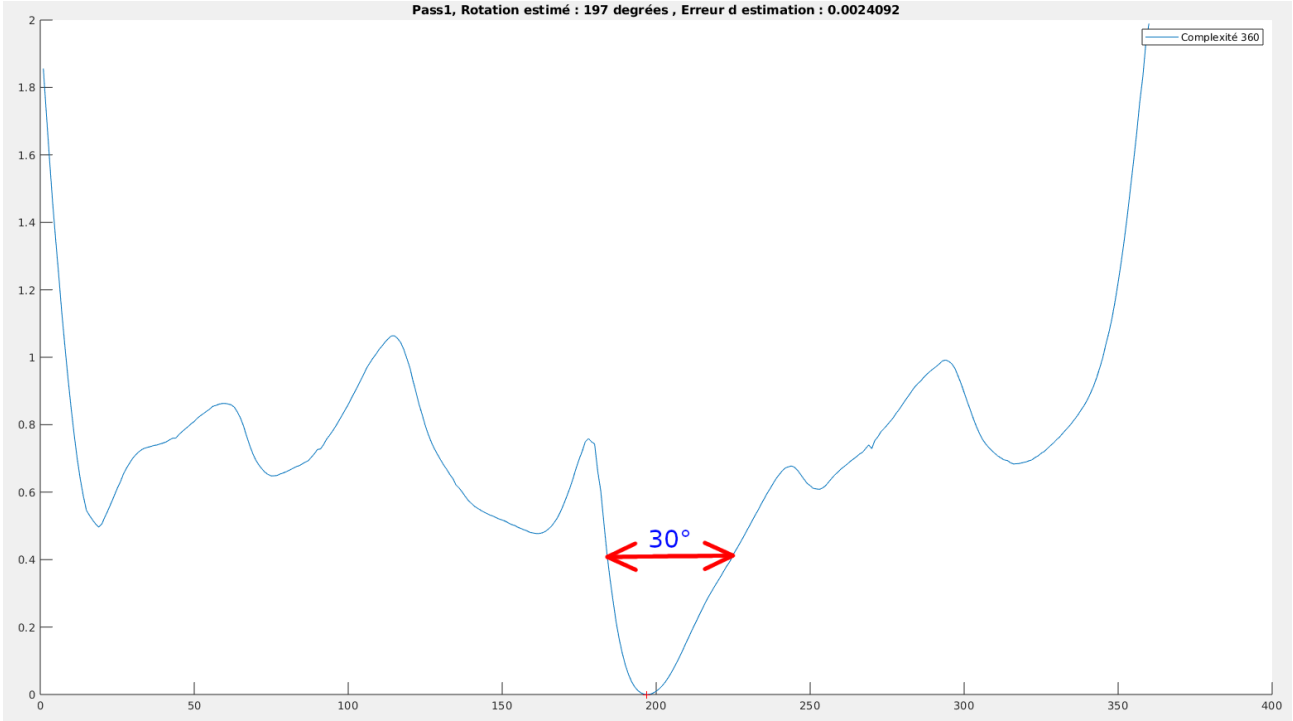


FIGURE 10

L'algorithme sera donc initialisé avec un **pas** à 30 degrés et la variable **pass** à 0, afin d'avoir au second tour un pas de 1 degré puis de 0.1, etc.
On s'assure ainsi de trouver toujours le minimum global.

La complexité de l'algorithme varie entre 136(5 pass) et 157(6 pass) en fonction de la précision nécessaire $\{10^{-4}, 10^{-5}\}$ pour retrouver l'image transformée.

Résultats et conclusions

On a de bons résultats avec l'algorithme d'optimisation, la précision de l'estimation de θ est optimale (égalité de R(x) et T(M(x)) avec une complexité raisonnable de plus ce résultat se généralise à l'ensemble des images et de l'ensemble des rotations recherchées sur la mesure corrélation croisée modifiée. Nos hypothèses semblent être vérifiées.

Par la suite du fait de la convexité des fonctions de coût j'ai tenté une optimisation par l'algorithme de la descente de gradient mais le fait d'avoir à $\delta\theta = \{10^{-4}, 10^{-5}\}$ des paliers ralentie la recherche du minimum de plus les hyperparamètres sont délicats, sans insité davantage il ne semble pas être valable pour toutes les valeurs recherchées. J'ai tous de même un résultat sur cet algorithme avec la mesure de corrélation croisée modifiée mais il n'est certainement pas représentatif et ne garantit pas de trouver le minimal.

Complexité : 452 Rotation estimé : 307.0918 degrées.

En conclusion nous avons compris l'intérêt des mesures de similarité dans ce type de problème, les algorithmes de recalage sont vus comme des problèmes d'optimisation. D'autres moyens d'augmenté l'efficience de l'algorithme, seraient de diminuer l'espace des caractéristiques, il serait intéressant aussi d'exploiter davantage la symétrie autour θ minimal.

On comprend bien que le problème peut devenir beaucoup plus difficile avec plusieurs paramètres dépendant sur des domaines de recherche plus étendus (translation, zoom, ...) avec le risque dans le cas ou l'image n'est pas représenté dans l'ensemble I d'être dans un minimum local.

Annexes

Hypothèse 1 : Allure des fonction invariant au images considéré

Les variations des 4 types de mesure de similarité en fonction de θ et des images de l'ensemble transformées à 70 degrés.

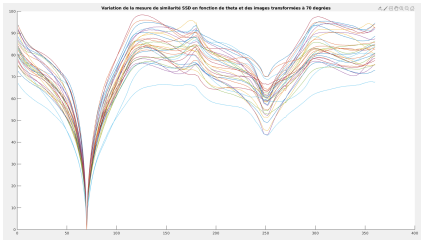


FIGURE 11 – SSD

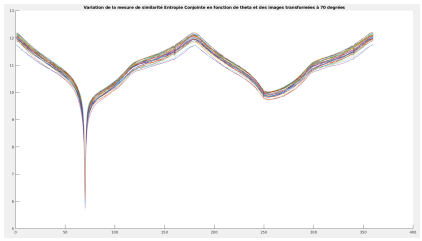


FIGURE 12 – Entropie conjointe

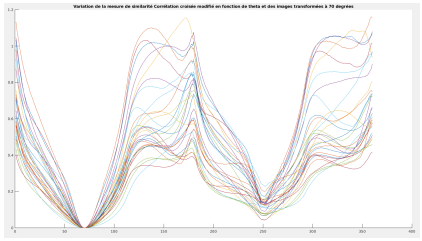


FIGURE 13 – Corrélation croisée modifié

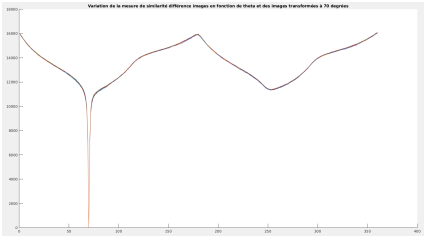


FIGURE 14 – Différence images

De manière graphique sur notre ensemble cette hypothèse est plus ou moins vérifiée en fonction du type de fonction.

Hypothèse 2 : L'allure des convergences vers la valeur minimale ne dépend pas de la valeur recherchée.
Malgré la troncature des images le "type" de convergence est la même.

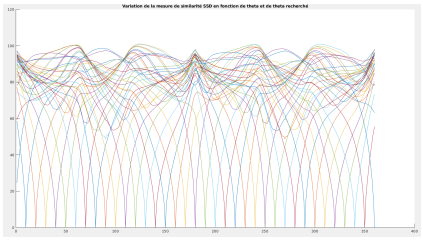


FIGURE 15 – SSD

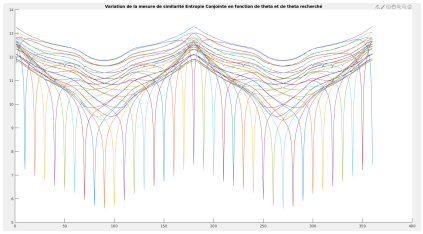


FIGURE 16 – Entropie conjointe

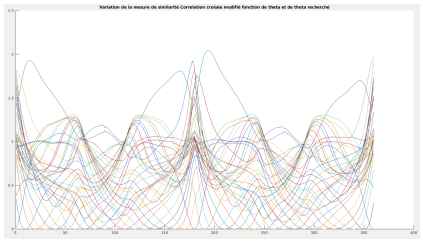


FIGURE 17 – Corrélation croisée modifié

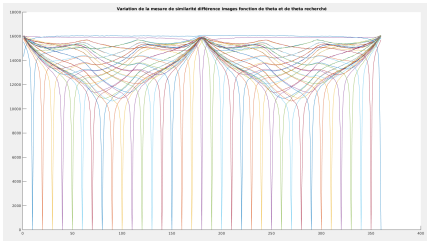


FIGURE 18 – Différence images