

# RAPPORT DE STAGE

## 3<sup>e</sup> ANNEE – EII

Stage du 11/06 au 31/07 2018

**INTITULE DU STAGE - EXTENSION D'UN  
LOGICIEL PEDAGOGIQUE DE TRAITEMENT  
D'IMAGES**

**Lieu du Stage**

Département EII – INSA Rennes

**Tuteur du Stage**


Julien HEULOT

**Correspondant pédagogique INSA**

Jean Gabriel COUSIN

## Remerciements :

Je tiens à remercier Luce Morin pour avoir proposé cette opportunité de stage au sein du département EII et pour les explications apportées sur certains algorithmes. Je remercie également Julien Heulot pour avoir répondu à nos questions et nous avoir guidé tout au long du stage. Enfin je remercie Bertrand Provot, stagiaire ayant travaillé avec moi, pour la coopération et la bonne humeur présente lors de ce stage.

A decorative red triangle is located in the bottom left corner of the page, pointing upwards and to the right.

## Sommaire

Remerciements :	2
Introduction.....	3
Mission .....	4
Organisation .....	4
Correction plantage.....	5
Histogramme .....	5
Seuillage .....	6
Transformée 8x8 et DCT 16x16 .....	7
Décomposition pyramidale .....	8
Correction des algorithmes .....	8
Transformée de Hough.....	8
Quantificateur de LloydMax.....	10
Amélioration et nouvelles fonctionnalités .....	11
Filtrage.....	11
Calcul d'entropie .....	13
Convertisseur double vers uChar .....	14
Création des tests.....	14
Conclusion .....	15
Résumé .....	16

## Introduction

J'ai réalisé mon stage au sein du laboratoire de recherche du département EII à l'INSA Rennes, parmi le personnel du département. Nous étions deux, Bertrand Provot étudiant en Informatique à l'INSA et moi-même à travailler sur le même projet, dans le bureau que nous avons partagé tout au long du stage. Nous restions malgré tout à proximité de nos interlocuteurs principaux qu'étaient Luce Morin et Julien Heulot lorsque nous avions besoin de conseils ou de faire le point sur nos avancées.

L'objectif était de travailler sur le développement du logiciel ImageINSA. ImageINSA est un outil de traitement de l'image développé par le département EII. Il est utilisé notamment dans les TP de traitement de l'image par les étudiants de l'INSA. Le logiciel a été développé par de nombreuses personnes aux fils des années. Cette année, l'objectif était dans un premier temps de résoudre les soucis importants du logiciel, comme les plantages et les bugs. Dans un second temps, apporter des améliorations à l'interface et corriger les algorithmes. Et enfin, simplifier le futur du développement en implémentant un de test automatique de l'application.

Ce rapport décrit le travail effectué lors de ce stage du 11 juin au 24 juillet 2018 puis du 20 août au 31 août 2018. Dans une première partie, on s'intéressera à l'organisation et la répartition du travail. Dans une seconde partie, on verra la correction des bugs importants, puis les améliorations apportées au logiciel ImageINSA et enfin la création des tests.

## Mission

### Organisation

ImageINSA est développé en C++ en utilisant la plateforme Qt et le système de compilation Cmake. L'application étant destinée à fonctionner aussi bien sur Windows que sur linux, nous avons donc travaillé chacun sur une plateforme. J'ai donc travaillé tout le long du stage sur Windows 7.

Le développement de ImageINSA est synchronisé sur Git. Julien nous a alors proposé d'utiliser le système des « issues » proposées par GitHub couplée à l'interface graphique Glo développée par GitKraken pour organiser et répartir nos tâches à effectuées.

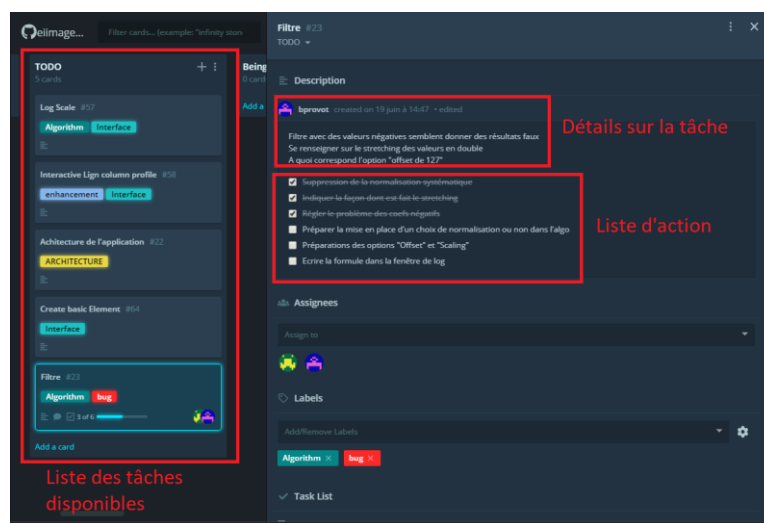


Figure 1 Description d'une tâche sur l'interface de Glo Board de GitKraken

Glo permet de présenter les tâches sous forme d'un tableau de tuile, chaque tuile représentant une tâche. On peut ainsi clairement et rapidement voir les tâches en cours, à faire et également ajouter des précisions et des commentaires pour garder une trace des problèmes et des choix qui ont été fait. Dans la pratique, je me concentrais sur une seule tâche à la fois, j'indiquais les problèmes rencontrés sur le bloc correspondant ainsi que mon avancée. Une fois la tâche effectuée j'indiquais la tâche comme faite et passait à la suivante. Ce système était également très pratique en réunion, il suffisait de regarder la colonne « terminée » ou « à faire » pour connaître l'avancement du projet et décrire nos actions et choix.

Bertrand Provot étant étudiant en Informatique et moi en EI, nous nous sommes naturellement répartis le travail de manière à ce qui ne se concentre que sur la partie interface et moi sur la partie algorithmique et traitement des images.

Au début du stage, on nous a fourni une liste de problèmes et améliorations à apporter à l'application. Cette liste étant des remarques observées par les utilisateurs, elles n'étaient pas très précises et ne concernait que la version Windows de ImageINSA.

Notre première tâche effectuée en parallèle sur Windows et linux a été de refaire les sujets de TP utilisant le logiciel afin de prendre en main le logiciel, de reproduire les bugs et plantages répertoriés ainsi que de découvrir d'autres bugs ou amélioration à apporter.

Nous avons alors pu établir un ordre des tâches à faire avec en priorité les plantages entrainant une fermeture de l'application, ensuite les bugs entrainant des résultats faux, et enfin les améliorations que nous pourrions apporter.

Une des améliorations importantes est d'ajouter la possibilité de tester la validité des résultats des algorithmes implémentés par rapport à une référence. J'ai alors également utilisé le logiciel Matlab pour générer des images de références à comparer aux résultats d'ImageNSA.

## Correction plantage

### Histogramme

L'histogramme représente la répartition des valeurs de pixels d'une image.

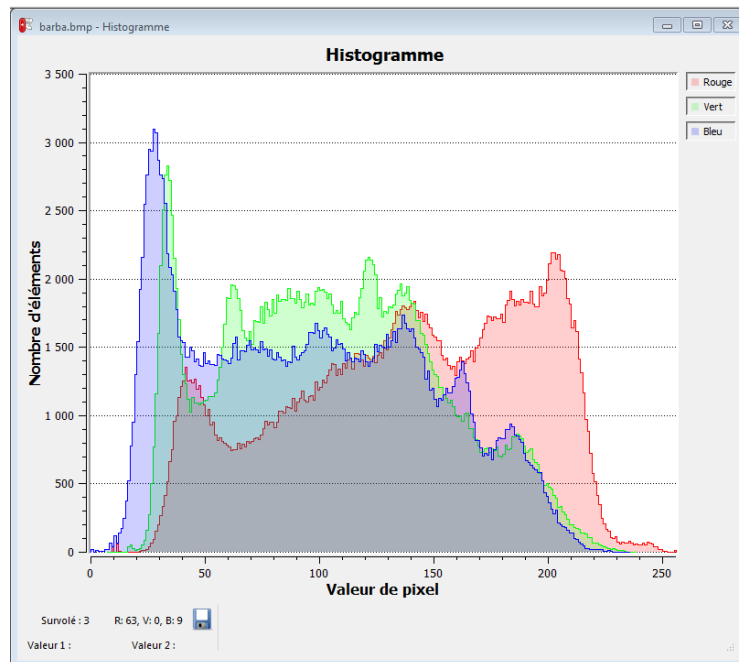


Figure 2 Exemple de visualisation d'histogramme dans ImageNSA (obtenu à partir de l'image barba). On peut y voir un histogramme par canal de couleur.

Dans l'application les histogrammes sont très utilisées, ils servent de bases à d'autres fonctionnalités (calcul d'entropie par exemple), la fiabilité de cette fonctionnalité est donc primordiale. L'utilisateur peut également afficher l'histogramme d'une image.

Les histogrammes sont modélisés par un tableau dont les valeurs des cases du tableau correspondent au nombre de pixels ayant pour valeurs l'indice de la case, ces indices allant pour une image classique de 0 à 255.

Le problème est que dans ImageNSA, on manipule à la fois des images dont les pixels sont compris entre 0 et 255 (dites de type uChar) et des images dont certains pixels ont des valeurs négatives et valeurs décimales (dites de type double).

Pour tracer l'histogramme d'une image double on utilise un pas de 1 en prenant toujours l'entier le plus proche. Il reste alors le problème dans le cas de pixels à valeurs négatives qu'on ne peut pas accéder à un indice négatif dans un tableau. C'est ce qui causait le plantage de l'application.

La solution que j'ai apportée est alors de modifier la façon dont est créé l'histogramme, on lui donne maintenant la valeur minimum, maximum des pixels de l'image. L'histogramme effectue alors le décalage d'index pour ne manipuler que des index positifs au moment d'accéder au tableau.

Problème :

Indice du tableau :		0	1	2	3	4	5	6	7						
Valeurs des pixels :	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
	Valeurs non-accessibles														

Solution apportée :

Indice du tableau :		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Valeurs des pixels :		-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
	Décalage d'indice	Valeurs accessibles														

Figure 3 Modification apportée à la structure de donnée de l'histogramme

Cette manipulation étant faite à l'intérieur de la classe Histogramme, les utilisations existantes des histogrammes n'ont pas eu à être modifiées.

## Seuillage

L'application plantait également lors de l'utilisation de l'outil de seuillage. Le seuillage consiste à transformer une image noir et blanc en une image binaire en fonction d'un seuil défini par l'utilisateur.

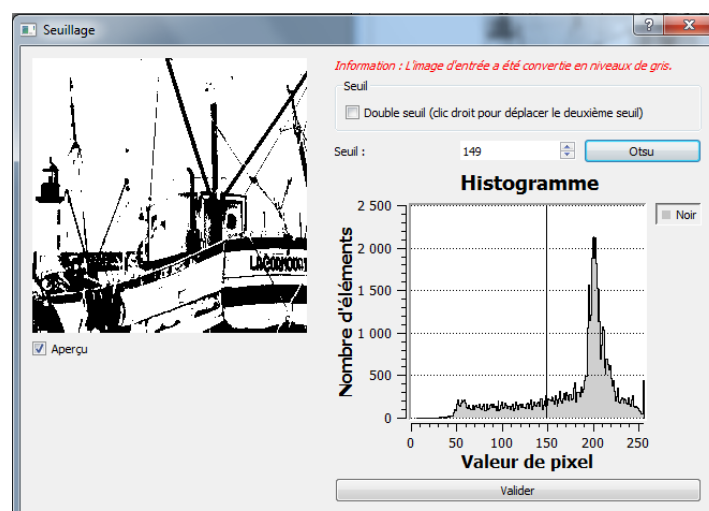


Figure 4 Interface permettant le seuillage d'une image dans ImageNSA

J'ai remarqué que l'application plantait lorsque l'on essayait beaucoup de seuils différents avec la prévisualisation. Après avoir utilisé Valgrind, j'ai pu localiser une fuite mémoire et la corriger. Une image de prévisualisation est générée à chaque fois que le seuil change et l'espace mémoire correspondant n'était pas libéré.

L'application ne plante désormais plus lors de l'application d'un seuil à une image.

## Transformée 8x8 et DCT 16x16

Ces transformées sont basées sur le produit matriciel d'une matrice de coefficient et de l'image subdivisée en morceau de la taille de la matrice de coefficient. Chaque bloc de l'image résultat est le résultat du produit entre la matrice de coefficient et le bloc correspondant de l'image source.

L'algorithme nécessite alors que les dimensions de l'image soient un multiple de la taille de la matrice de coefficient, hors rien n'était fait si ce n'est pas le cas on accède alors à des valeurs en dehors de l'image ce qui entraîne un plantage de l'application.

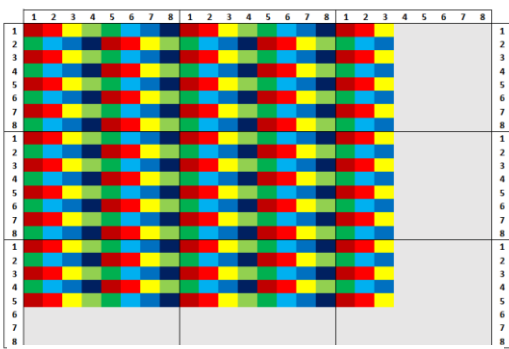


Figure 5 Exemple de découpe en bloc de 8x8 d'une image de 19x21. On peut remarquer qu'une partie des blocs est hors de l'image.

J'ai donc dans un premier temps résolu le problème en tronquant la matrice de coefficient au niveau de la limite de l'image.

Cette solution permettait de ne plus avoir de crash lors de l'exécution mais les résultats obtenus n'étaient pas corrects, dans ce genre de cas on applique une politique aux bords, consistant par exemple à recopier la dernière ligne.

J'ai donc modifié l'algorithme pour créer une image résultat de la même taille que l'image source mais agrandie jusqu'au prochain multiple. J'ai choisi cette méthode pour deux raisons, d'une part d'un point de vue pédagogique on comprend bien l'opération effectuée, d'autre part pour appliquer la transformée inverse il est nécessaire d'avoir l'image résultat complète.

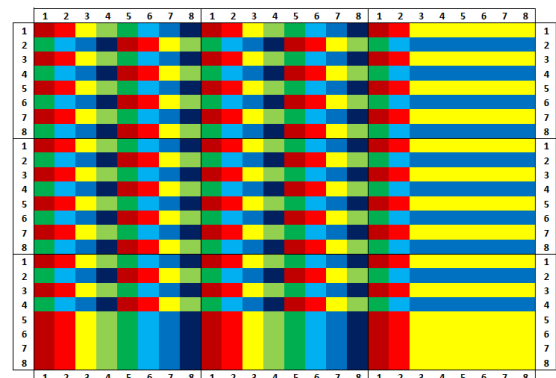


Figure 6 Image agrandie jusqu'au prochain bloc en recopiant la dernière ligne



## Décomposition pyramidale

La décomposition pyramidale consiste en la représentation d'une même image sur plusieurs niveau en appliquant un filtre à l'image et en divisant à chaque niveau la taille de l'image par deux.

L'utilisation du filtre QMF lors de la décomposition pyramidale entraînait un plantage du logiciel. Les filtres sont définis par des structures et des constantes dans le code. Le tableau contenant ces structures était donné octet par octet en hexadécimale.

```
const uint8_t tp6_filter_file_data[] = {0x74, 0x72, 0x69, 0x61, 0x6e, 0x67, 0x75, 0x6c, 0x61, ... , 0x00, 0x00, 0x04, 0x00, 0x00};
```

*Figure 7 Ancienne description de la composition des filtres*

J'ai donc décodé pour retrouver les valeurs de chaque champ des structures. Je me suis aperçu qu'il y avait une erreur dans la définition du filtre QMF, deux octets manquants.

```
54 struct Filtre {
55     char nom_f[30];           //!< Name of the filter
56     float coeff_f[10];       //!< Coefficient of the filter
57     int taille_f;            //!< Size of the filter
58 };
```

*Figure 8 Déclaration de la structure des filtres*

Ayant toutes les structures décodées, j'ai comparé les valeurs avec les valeurs théoriques des filtres en question. Pour cela, j'ai cherché un cours expliquant la décomposition et le fonctionnement de ces filtres. Une fois fait, j'ai entrepris de simplifier le code en réécrivant les structures de façon littérale, pour faciliter le travail du prochain contributeur. J'ai également simplifié et optimisé la façon dont sont manipulés les filtres, en rajoutant des cas par défaut si le filtre n'est pas trouvé et en limitant le nombre de fois où l'on parcourt toute la liste de filtre.

## Correction des algorithmes

### Transformée de Hough

La transformée de Hough est une méthode de détection des lignes droites dans une image. Pour chaque point blanc de l'image on cherche toutes les droites passant par ce point et on les répertorie dans l'espace de Hough. Les points d'accumulation dans l'espace de Hough correspondent alors aux droites passant par plusieurs points de l'image source.

Les corrections à apporter à l'implémentation de la transformée de Hough étaient les suivantes :

- S'assurer que les différents pas pour rho et theta étaient bien utilisés pour la transformée de Hough inverse.
- Homogénéiser les axes utilisés avec ceux utilisés dans le cours. (x, y pour l'image source et rho, theta dans l'espace de Hough)

Pour effectuer ces modifications j'ai d'abord dû comprendre la transformée de Hough et l'implémentation actuelle. Pour cela, j'ai eu accès aux cours et aux explications de Luce Morin.

J'ai donc pu effectuer le changement de repère en plaçant l'origine dans le coin haut-gauche et en inversant les axes x, y par rapport à ceux utiliser naturellement dans l'image.

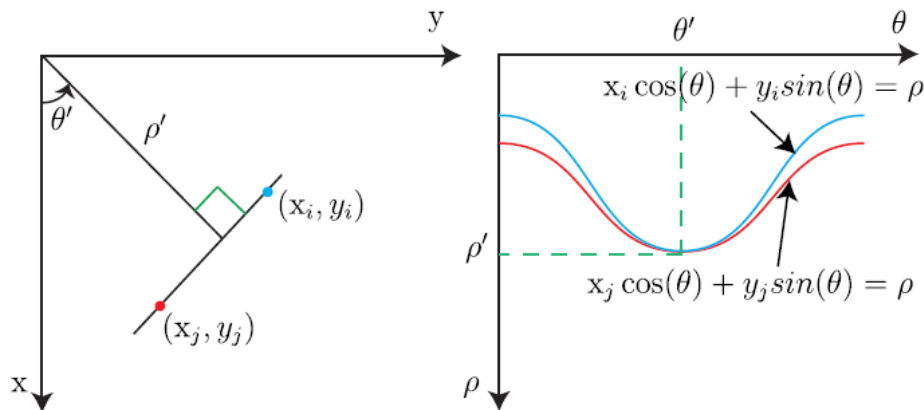


Figure 9 Nouveau repère utilisé pour l'image et l'espace de Hough

Pour l'espace de Hough on garde les angles compris entre  $|\theta| < \pi/2$  et  $|\rho| < \text{diagonale de l'image}$ . Les seules droites nécessitant un traitement sont les droites ayant  $|\theta| \geq \pi/2$ , dans ce cas on a :

$$\text{nouv\_theta} = \theta - \pi \quad \text{et} \quad \text{nouv\_rho} = -\rho$$

Pour le pas, il est bien pris en compte lors de la transformée de Hough inverse. En effet la taille de l'image dans l'espace de Hough dépend des dimensions de l'image originale et des pas utilisés pour rho et theta. En connaissant les dimensions de l'image à reconstruire et celles de la transformée de Hough, on peut retrouver les pas utilisés.

Cependant, une erreur était présente dans le code : diagonale de l'image = largeur \*  $\sqrt{2}$ . La transformée de Hough inverse ne fonctionnait alors que sur des images carrées. J'ai donc corrigé par la formule : diagonale de l'image =  $\sqrt{(\text{largeur}^2 + \text{hauteur}^2)}$ .

Après vérification sur des images de dimensions différentes en utilisant des pas différents, les résultats sont cohérents avec la théorie.

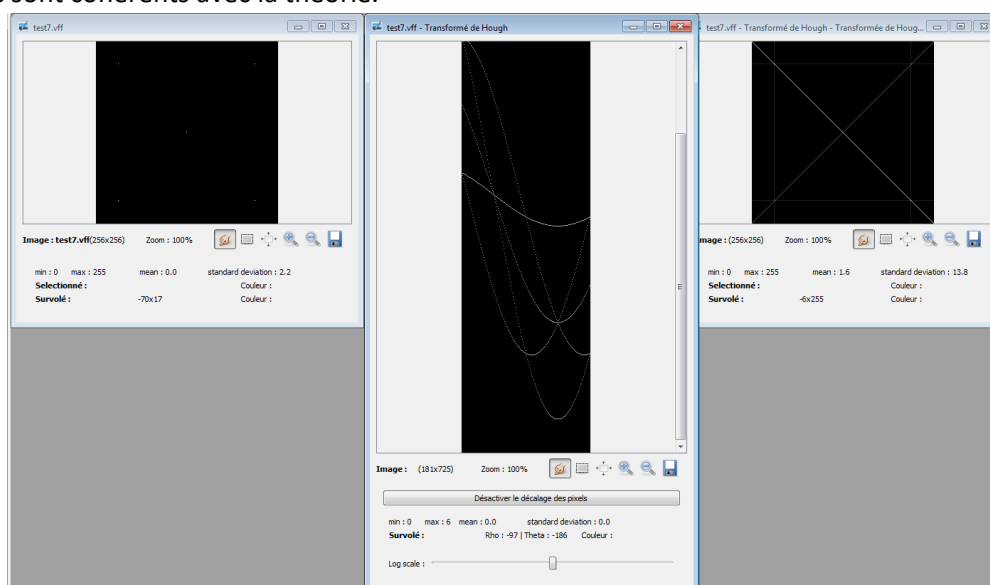


Figure 10 Exemple de transformée et de transformée inverse après modification. (Image originale à gauche, transformée de Hough au centre et transformée inverse à droite)

## Quantificateur de LloydMax

La quantification consiste à réduire le nombre de niveaux de couleurs dans une image. On utilise pour cela une loi de quantification permettant d'effectuer la transformation.

LloydMax est un algorithme permettant de trouver le quantificateur optimal pour une image, c'est-à-dire le quantificateur minimisant la distorsion, pour cela on place les seuils de décisions au milieu des intervalles des valeurs et les valeurs sont les barycentres entre les seuils de décisions.

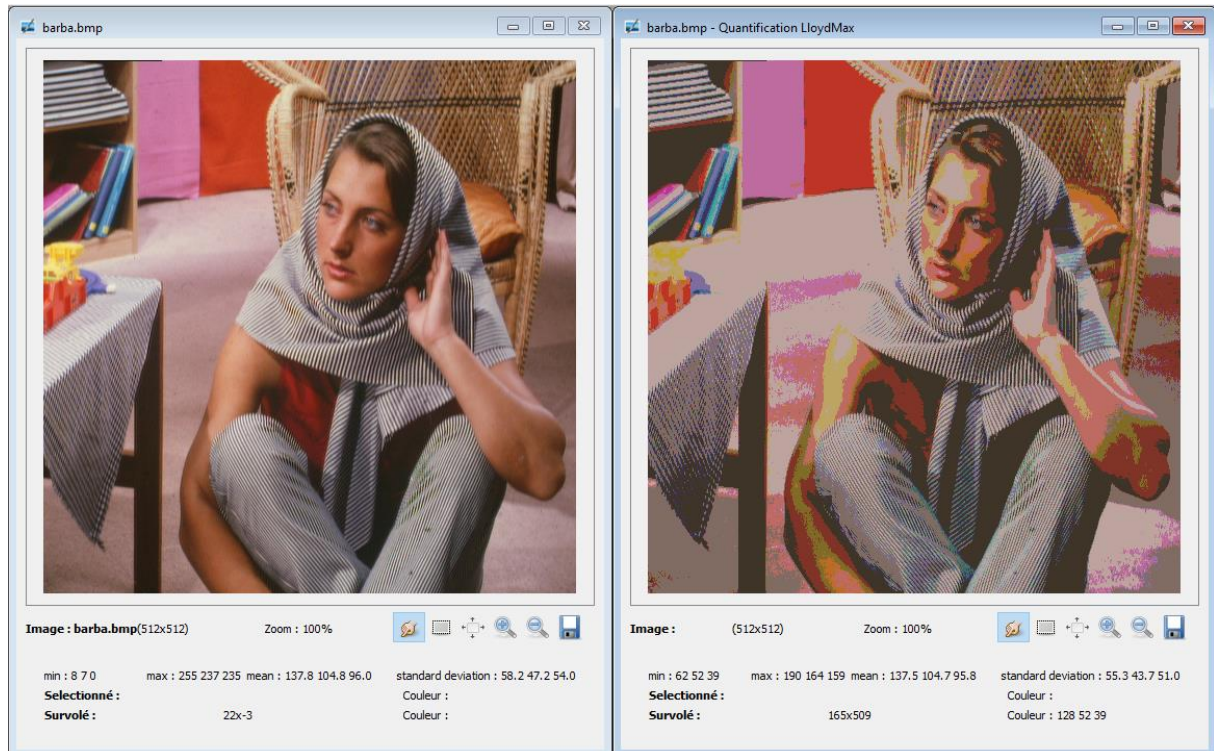


Figure 10 Exemple de quantification de utilisant un quantificateur de LloydMax dans ImageINSA

J'ai donc implémenté l'algorithme de LloydMax en utilisant l'histogramme de l'image pour calculer la position des barycentres. L'algorithme implémenté est un algorithme itératif, on calcule alternativement la position des seuils de décision à partir des valeurs et les valeurs à partir des seuils de décision. Le problème que j'ai alors rencontré consistait à savoir quand arrêter la boucle.

S'il n'y a pas de pixels dans la plage où l'on souhaite calculer le barycentre, alors pour éviter une division on place simplement le barycentre au milieu de l'intervalle.

J'ai choisi d'arrêter la boucle quand le système n'évolue plus ou très peu, c'est-à-dire quand la moyenne des différences entre les anciens seuils et les nouveaux est inférieure à 1. Par sécurité et pour garantir la rapidité du logiciel nombre de boucle est également limité à 50. L'algorithme converge en générale bien avant. Il est maintenant possible d'appliquer une quantification de LloydMax sur toutes images via ImageINSA.

## Amélioration et nouvelles fonctionnalités

### Filtrage

Une importante amélioration concernait les applications de filtres à des images. Dans ImageINSA l'application de filtre par l'utilisateur se fait via une fenêtre de dialogue, l'objectif de ces améliorations était de clarifier les résultats donnés par le logiciel ainsi que d'offrir plus de libertés à l'utilisateur. Il a donc été décidé d'ajouter la possibilité de normaliser ou non les coefficients du filtre à appliquer, d'offrir plus de choix à l'utilisateur dans la façon d'obtenir le résultat (Image en uChar ou double, ajout d'un offset), ainsi que d'afficher les images intermédiaires résultat de chaque composante pour les filtres multiples.

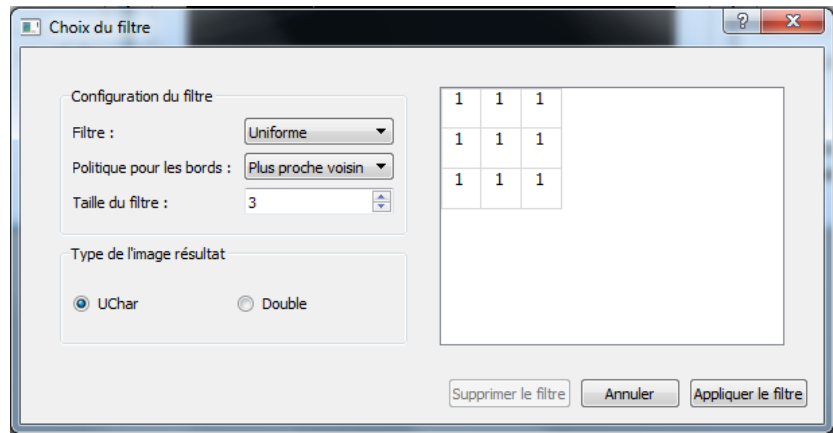


Figure 11 Ancienne version de l'interface de sélection d'un filtre

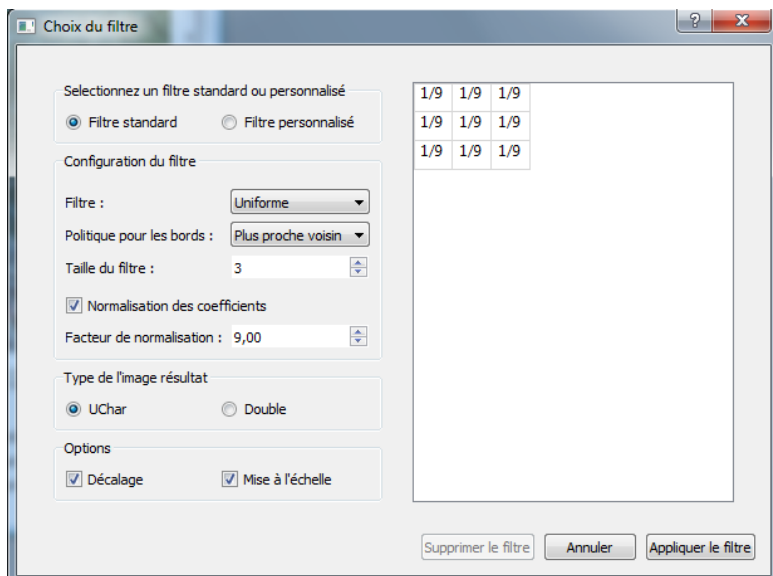


Figure 12 Nouvelle version de l'interface de sélection d'un filtre

Bertrand Provot a développé l'interface de cette nouvelle fenêtre de dialogue, nous avons donc dû collaborer. Je lui ai expliqué l'intérêt de ces nouveaux boutons et nous nous sommes mis d'accord sur le comportement de chacun de boutons en fonction de ce qui est possible et de ce qui était nécessaire d'un point de vue fonctionnalité.

### Normalisation

L'ancienne version de l'application appliquait une normalisation systématique des coefficients, mais rien n'indiquait à l'utilisateur que cette normalisation existait. J'ai donc ajouté un paramètre au moment de l'application du filtre activant la division des coefficients du filtre par un facteur donné.

Le facteur de normalisation est modifiable par l'utilisateur, mais sa valeur par défaut est calculée à partir des coefficients du filtre. Il est égal à la valeur absolue de la somme des coefficients du filtre. Si la somme des coefficients est égale à 0 alors le facteur de normalisation est égal à la somme des coefficients positifs.

Dans le cas de filtre multiple, on a qu'un seul facteur de normalisation pour tous les filtres il est égal au maximum des facteurs théoriques de chaque filtre. Ce choix a été fait pour simplifier

l'utilisation de l'outil de normalisation et car dans le cas classique les facteurs de normalisations sont -  
Choix de la forme du résultat :

L'ancienne version proposait deux choix pour l'obtention du résultat : UChar et Double. Ils décrivent la forme dont seront données les valeurs des pixels de l'image résultat. Pour plus de clarté, j'ai fait en sorte d'écrire la formule utilisée pour la conversion dans une chaîne de caractère qui pourra être affichée dans la fenêtre de log.

Il a également été décidé d'ajouter des options pour choisir le type conversion à appliquer. Dans le cas d'un résultat double le résultat est le résultat brut de l'application du filtre sous forme de double.

Dans le cas d'un résultat en uChar deux options peuvent être cochées :

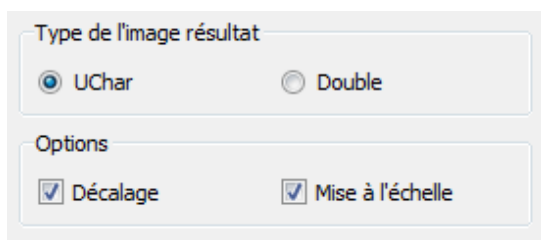


Figure 13 Extrait de l'interface de filtrage, montrant les options d'obtention du résultat

Figure 13: Extrait de l'interface de filtrage. La section 'Type de l'image résultat' contient deux boutons radio : 'UChar' (sélectionné) et 'Double'. La section 'Options' contient deux cases à cocher : 'Décalage' (cochée) et 'Mise à l'échelle' (cochée).

Figure 13 Extrait de l'interface de filtrage, montrant les options d'obtention du résultat

- Si aucune n'est cochée le résultat est simple tronqué, on ne garde que les valeurs comprises entre 0 et 255. Les valeurs plus grandes que 255 sont ramenées à 255, et celles négatives à 0.
- Si « décalage » (offset) est cochée alors un offset de 127 est ajouté à l'image si besoin. (Uniquement si le résultat contient des valeurs négatives)
- Si « mise à l'échelle » est cochée on garde que les valeurs positives et on les répartie proportionnellement entre 0 et 255
- Si « mise à l'échelle » et « décalage » sont cochés, on centre les valeurs autour de 127 et on répartie les valeurs négatives entre 0 et 127 et positives entre 127 et 255.

L'application du filtre rend une image en double, pour réaliser ces actions j'ai donc créé quatre nouveaux convertisseur, ces fonctions prennent en entrée une image double et retourne une image en uChar sur laquelle ont été appliquées les transformations précédentes.

Ces convertisseurs sont également réutilisables par d'autre partie de l'application.

### *Images intermédiaires*

Une demande d'amélioration concernait les filtres multiples, on souhaitait plus d'information et l'affichage des résultats de chaque composante du filtre.



J'ai donc fait en sorte d'obtenir chacune de ces images en rendant un vecteur contenant chacune des images des filtres. Une image résultat permet d'avoir un aperçu global du résultat du filtre. J'ai fait le choix d'utiliser la racine de la somme des carrés de chaque image intermédiaire, ainsi pour un filtre donnant le gradient d'une image on obtient en l'image finale la norme du gradient dans l'image.

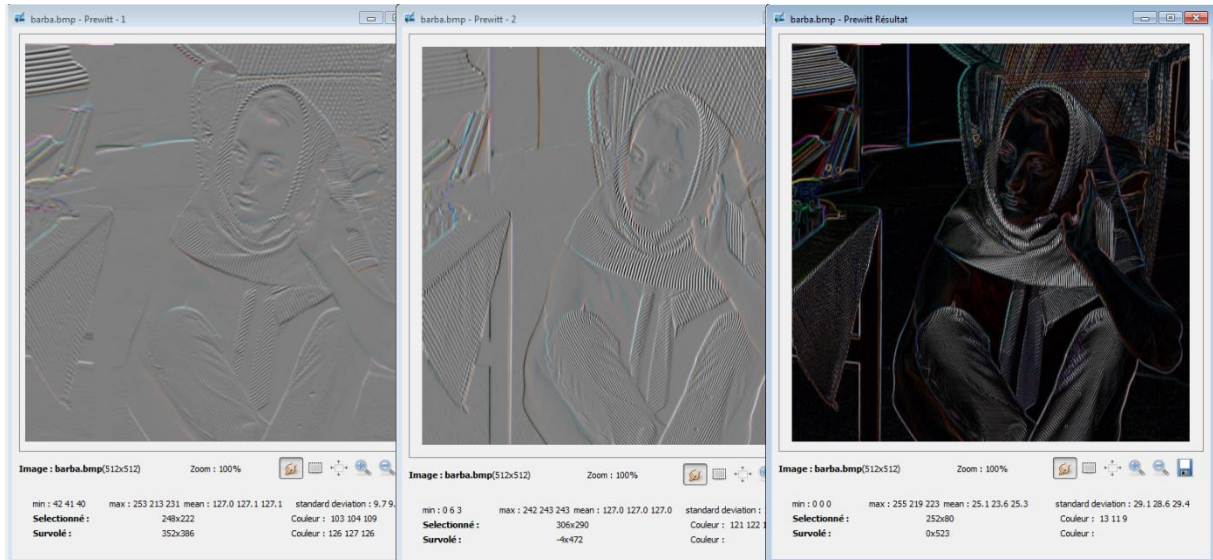


Figure 14 Images intermédiaires de résultat d'un filtre de Prewitt à gauche et l'image correspondant au gradient de l'image à droite

## Calcul d'entropie

ImageINSA permet de calculer l'entropie d'une image, le résultat est affiché dans la fenêtre de log. Certaines opérations affichent également automatiquement le résultat du calcul d'entropie dans la fenêtre de log. Le problème était que pour une même image les deux méthodes ne donnaient pas toujours le même résultat. Le problème venait du fait qu'à chaque fois qu'un calcul d'entropie est nécessaire dans les algorithmes le calcul était recodé. Ce n'est pas une bonne pratique car si on doit corriger une erreur de calcul il faut la corriger partout où l'entropie est calculée.

Pour remédier à cela, j'ai intégré à la classe Image une méthode `getEntropy()` qui retourne l'entropie calculé sur l'image courante. Ainsi, le calcul est centralisé et accessible pour n'importe quelle image dans l'application.

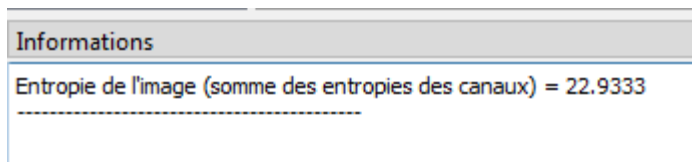


Figure 15 Exemple d'affichage du résultat du calcul de l'entropie

J'ai également remplacé tous les endroits où un calcul d'entropie était effectué par un appel à cette fonction, ainsi tous les résultats sont cohérents et les modifications éventuelles seront plus simples à apporter.

## Convertisseur double vers uChar

Nous avons décidé de développer un outil permettant de transformer une image à valeur double en image à valeur en uChar, celui-ci devrait servir à continuer à faire des opérations sur une image ayant reçu un traitement renvoyant des valeurs doubles.

Afin de garder un maximum d'information et en fonction des utilisations, j'ai fait en sorte de proposer plusieurs options de conversions on y retrouve les mêmes que celles présentes dans filtre (décalage et offset), on réutilise alors les convertisseurs déjà implémenter. Il y a également une autre option « normalisation » qui permet d'obtenir un meilleur contraste en remplaçant la valeur double minimum à 0 et la valeur maximum double à 255 dans l'image en Uchar, tout en répartissant les valeurs intermédiaires.

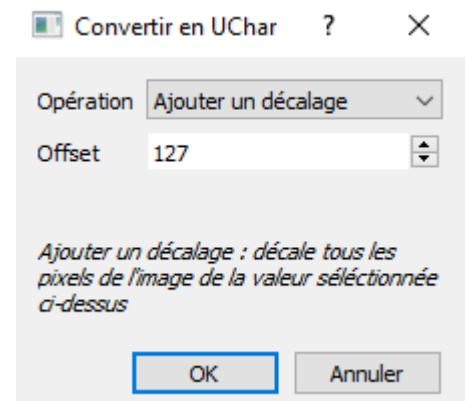


Figure 16 Interface du convertisseur double vers uChar de ImageINSA

## Création des tests

Afin de s'assurer du bon fonctionnement de l'ensemble de l'application, il nous a été suggérer de créer un système de test des fonctionnalités d'ImageINSA. Ces tests serviront également pour les futurs contributeurs. Bertrand Provot s'est occupé de développer le programme de test ainsi que les tests de l'interface, tant dis que je me suis concentré la validation des résultats de chaque opération.

Pour vérifier la validité du résultat d'une opération, j'ai créé des images de référence en utilisant les fonctions de traitement de l'image du logiciel Matlab. Il fallait s'assurer que les opérations effectuées sur Matlab étaient bien les mêmes que celle d'ImageINSA. Pour cela je me suis intéressé à chaque opération et à son fonctionnement précis.

J'ai donc dû lire la documentation de Matlab pour trouver la fonction équivalente et/ou les bonnes options à utiliser. J'ai également créé des fonctions Matlab servant de surcouche aux fonctions Matlab afin d'utiliser les bons paramètres.

En utilisant ces fonctions et les images utilisées en tps, j'ai alors créée une banque d'image résultat de chaque algorithme via Matlab qui pourra être utilisées comme référence lors des comparaisons avec les images générées par le logiciel. Ces images sont de différents types, couleur, noir et blanc, carré, rectangle, taille étant une puissance de 2 ou non afin de détecter d'éventuels problèmes.

## Conclusion

Lors de ce stage au département EII, j'ai l'occasion de travailler au développement d'un logiciel de traitement de l'image. Les objectifs de ce stage étaient de corriger tous les bugs connus, de modifier les algorithmes en place afin de corriger leur fonctionnement ou de le réadapter aux besoins des cours. L'objectif était également de reprendre l'architecture du logiciel pour simplifier son fonctionnement et la reprise du projet pour les prochains développeurs ainsi que de développer la plateforme de test.

La stabilité du logiciel a été grandement augmentée, il n'existe plus de plantage connu dans la version sortie après le stage. Les principales modifications apportées aux algorithmes ont été implémentées. J'ai créé des images de référence et des scripts Matlab imitant les algorithmes d'ImageNSA. La plateforme de test n'a pas pu être complètement terminée mais les images et les scripts pourront être réutilisés.

Le logiciel va continuer d'évoluer d'année en année, c'est pourquoi nous avons laissé une liste d'idée d'évolution à apporter sur la Glo Board (également trouvable dans les issues Git). J'ai également pris soin de commenter le code ainsi que de compléter les différents guides et documentations qui pourront aider les prochains contributeurs à prendre en main le logiciel.

Ce stage a été pour moi l'occasion d'améliorer ma capacité à développer en C++ ainsi que d'implémenter des algorithmes complexes. Ce fut également l'occasion d'obtenir une première expérience de travail sur un logiciel existant et ayant été repris de nombreuses fois par différents développeurs. Le projet sera également continué après moi, il faut donc garder cela à l'esprit quand on développe des fonctionnalités. Ce stage m'a également apporté des connaissances en traitement de l'image et traitement du signal, en effet j'ai travaillé et implémenté des algorithmes nécessitant des notions que j'ai dû acquérir pendant le stage.



## Résumé

### Extension d'un logiciel pédagogique de traitement d'images

**Mots clés :** Image, Traitement de l'image, C++, Développement logiciel, Algorithme

Ce rapport présente mon stage au département EII de l'INSA de Rennes, école d'ingénieur où je fais mes études en Electronique et Informatique Industrielle. J'y présente mes actions effectuées sur le développement du logiciel ImageINSA, logiciel pédagogique de traitement de l'image développé à l'INSA et utilisé dans le cadre des cours. L'objectif de ce stage était de corriger les bugs faisant suite à la refonte du logiciel et des nombreux contributeurs ayant travaillé sur le logiciel. Il était également prévu de corriger quelques algorithmes et d'apporter des améliorations notamment concernant l'interface. Nous étions deux stagiaires sur ce travail, Bertrand Provot étudiant en Informatique à L'INSA Rennes m'a épaulé, nous avons dû nous répartir le travail, il s'est occupé du développement de l'interface et moi de la partie algorithmique. Ce rapport présente les modifications apportées à l'implémentation de la transformée de Hough ainsi que l'implémentation du quantificateur de LloydMax et la création des images de tests.

### Extension of an educational image processing software

**Key Words:** Image, Image processing, C++, Software Development, Algorithm

This report presents my internship at the EII department in INSA Rennes, where I study Electronic and Industrial Computer Science. I worked on ImageINSA, which is an image processing software used for educational purpose. The objective was to correct every known issues that appeared since the remake of the application and lot of developers has worked on it. Another objective was to improve some functionality, either regarding algorithm and user interface. We were two trainee working on this project, Bertrand Provot computer science student at INSA Rennes helped me, we had to split task, so Bertrand worked more on the user interface and I worked on algorithms. This report present the modification provided on Hough Transform, LloydMax quantifier and also the way the reference image and Matlab function was conceived.

## **INSA Rennes**

20 Avenue des Buttes de Coësmes  
CS 70839  
35708 Rennes Cedex 7

Tél. +33 (0) 2 23 23 82 00

Fax +33 (0) 2 23 23 83 96

[www.insa-rennes.fr](http://www.insa-rennes.fr)

**INSA**

UNIVERSITE  
BRETAGNE  
LOIRE

**Cti**  
Commission  
des Titres d'Ingénieur

