Développement d'un outil de segmentation et de classification semi-automatique de pierres

Pierre-Alban 6-02-1992 Hugueny

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

ÉCOLE SUPÉRIEURE DES GÉOMÈTRES ET TOPOGRAPHES

MÉMOIRE

présenté en vue d'obtenir

le DIPLÔME D'INGÉNIEUR CNAM

Spécialité : Géomètre et Topographe

par

Pierre-Alban HUGUENY

Développement d'un outil de segmentation et de classification semi-automatique de pierres

Soutenu le 09 juillet 2014

JURY

PRÉSIDENT : **M. Rani EL MÉOUCHE**

MEMBRES :
M. Hugo MEUNIER, maître de stage
Mme Élisabeth SIMONETTO, maître de stage
M. Jean-Michel FOLLIN, professeur référent
M. Yannick ARDOUIN
M. Éric LABERGERIE
M. Yvan LIGOUT
M. Cyril MICHON
M. Laurent POLIDORI

Remerciements

Tout d'abord, je tenais à remercier une de mes commanditaires et professeure de l'école : Mme Elisabeth Simonetto. Pour son dévouement envers le sujet, pour le temps qu'elle a pu me consacrer bien que je ne sois pas facile à diriger, je la remercie de tout mon cœur. Simplement, sans elle, le résultat actuel n'aurait jamais été atteint. Merci.

Pour ses critiques et ses choix judicieux, pour son aide professionnelle en tant que professeur référent, je tiens à remercier M. Jean-Michel Follin. Il m'a permis d'accéder au résultat par ses remarques mais non moins très fructifiantes. Garder le bon cap durant toutes ces heures passées sur ce sujet est quelque chose de très important et grâce à mon professeur référent, j'ai le plaisir de réussir.

D'autre part, je tenais à remercier le Centre Allonnais de Prospection et de Recherches Archéologiques et plus précisément M. Hugo Meunier. Ce dernier a cru en moi (et sans fausse modestie, cela n'a pas de prix), il m'a permis d'exprimer et d'approfondir, par l'intermédiaire de ce travail, le caractère d'ingénieur que j'ai pu me forger au cours de ces 3 ans à l'ESGT.

Pour terminer, je tenais à remercier l'ESGT pour m'avoir accueilli en son sein durant toutes ces années ainsi que tous les professeurs ayant participé de près ou de loin à la bonne réalisation de ce projet. Ce dernier fut une expérience des plus enrichissantes et formatrices. Je n'aurais jamais pu acquérir les bases de mon futur métier de Géomètre-Expert sans toutes ces personnes.

Sans paraître banal et redondant, je tiens vraiment à remercier tous les encadrants, professeurs et administratifs de l'école, pour le temps qu'ils ont pu m'accorder jusqu'à ce jour. Je suis fier de ce que j'ai fait et de ce que je suis devenu, et c'est grâce à eux.

Merci.

Table des matières :

1. **Introduction**

Dans le domaine de l'archéologie, on compte diverses disciplines et champs d'intérêt comme les artefacts (outils, bijoux..), la botanique ou encore le bâti. C'est ce dernier qui nous concerne. L'archéologie du bâti est une méthode d'analyse des constructions (plus ou moins anciennes) qui consiste à étudier l'évolution d'un bâtiment, les matériaux utilisés et les techniques de construction. L'étude de ces derniers se décompose en la détermination de points précis comme le lieu de leur implantation, la matière des ressources utilisées, leur qualité, leur taille, leur utilisation … tant d'indices qui permettent à un archéologue d'émettre des hypothèses quant à l'histoire du lieu (endroit d'implantation et bâti).

Plus particulièrement, l'archéologue, dans sa recherche, va étudier les différentes étapes de construction et reconstruction de l'ossature de l'édifice à l'aide d'indices et de méthodes de datation.

Nous retrouvons, parmi les étapes de cette étude archéologique, le relevé. Un relevé préalable à toute recherche est indispensable. L'édifice étudié passe alors d'un objet matériel à une image. L'archéologue acquiert ses mesures terrain et peut ensuite appliquer à ces valeurs numériques ses recherches et conclusions.

Auparavant, les archéologues devaient se contenter de dessiner à main levée ce qu'ils observaient sur les façades (intérieures et extérieures) des bâtis. Ils pouvaient s'aider d'appareils photo numériques, de pantographes, etc. Depuis quelques dizaines d'années, ère des nouvelles technologies et des ordinateurs, les archéologues sont accompagnés lors de leurs acquisitions de données par de nouveaux outils. On retrouve parmi ceux-ci le tachéomètre (ou théodolite laser), la prise d'orthophotographie mais aussi le scanner laser 3D. Malheureusement, même si cela améliore considérablement la qualité des résultats (précisions millimétriques), les temps de traitement et d'acquisition des données sont toujours conséquents (dessin et qualification des pierres de façade : environ 5000 pierres / semaine – d'après le Centre Allonnais de Prospection et de Recherches Archéologiques (CAPRA)).

N'oublions pas ce chiffre puisque lors des étapes d'acquisition et de traitement des données, c'est la phase de dessin des contours des pierres, plutôt rébarbative, qui prend le plus de temps. A notre connaissance, il n'existe pas de logiciel dédié à la détection et la qualification des pierres et briques de façades.

Notre souhait, en collaboration avec le CAPRA, est donc de remédier à cette lacune : en établissant un outil de segmentation et classification de murs de façade en pierres et briques. En considérant que rien n'a déjà été établi, est ce possible ? Avec quel degré de liberté ? Quels résultats ? Sous quelle forme ? La machine ne pouvant remplacer l'homme et sa perception, nous savons que le résultat ne sera pas parfait. L'utilisateur devra donc pouvoir corriger ces résultats interactivement. L'outil sera alors semi-automatique. Néanmoins, les utilisateurs ne seront pas férus de traitement d'image, les choix des paramètres devront être vus en conséquence. Concernant directement les besoins du CAPRA, ces derniers utilisent comme source de leurs travaux des orthophotos au format TIFF[1] de tailles (en Méga-octet : Mo) très diverses, souvent très pixellisées (pouvant dépasser les 50 000 x 50 000px). Ces images étant géoréférencées, le résultat devra être de même. Actuellement, le CAPRA utilise le pack logiciel payant de la société ESRI, ArcGIS. Les archéologues importent dans ce logiciel leurs orthophotos puis dessinent, à l'aide des outils de dessin à disposition dans ArcMap, les contours des pierres, briques, etc. Il est alors produit un résultat de format vectoriel. Ces résultats ainsi produits par le CAPRA seront nommés « vérité terrain » par la suite. Concernant les bâtiments que traitent ou pourraient traiter le CAPRA, il en existe de nombreux, de tailles, formes, teintes, matériaux, etc... différents. Un même bâtiment peut comporter de multiples éléments. En accord avec le CAPRA, l'outil produit devra en priorité fonctionner pour les façades de la cathédrale du MANS.

[1] Tagged Image File Format

Si, à cette heure, il n'a encore rien été produit concernant notre démarche, c'est qu'il existe bien des problématiques (tout du moins temporaires). Plus particulièrement, la diversité (en terme de couleurs, formes, tailles...) des images pouvant être traitées est si vaste que cela semble à première vue un enjeu.

Après un temps de recherche, une liste de certaines bibliothèques d'algorithme de traitement d'image est établie. Nous étudierons lors d'une première partie les divers algorithmes de segmentation disponibles dans ces bibliothèques. Après avoir appréhendé ces méthodes, nous établirons un protocole d'expérimentation pour tester leurs possibilités et limites. Nous verrons que dans nos expérimentations, les résultats seront toujours exportés en raster.

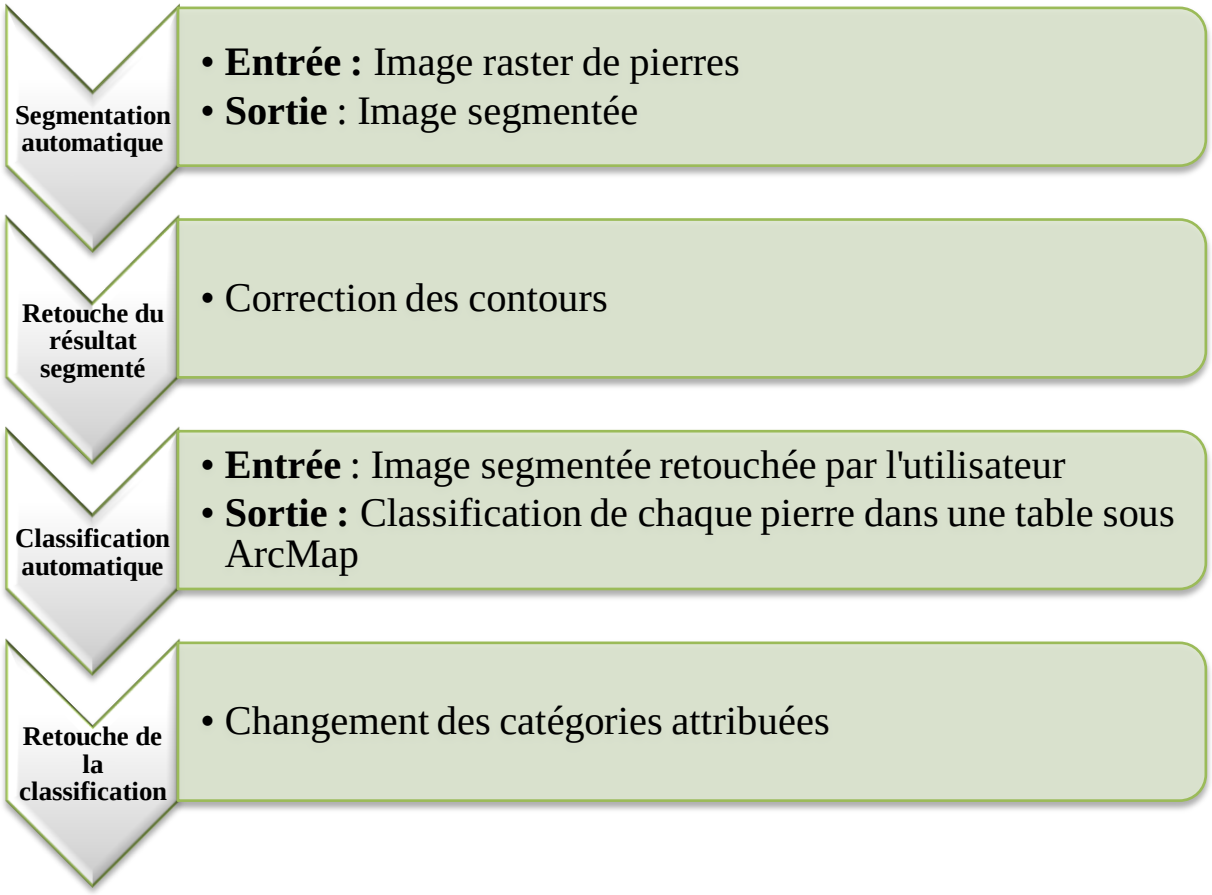
Chronologiquement, le second chapitre se portera tout naturellement sur la transformation du raster de sortie en un fichier vectoriel comprenant les limites établies automatiquement dans la première partie de notre outil. De plus, nous décrirons les méthodes qui s'offrent à nous pour permettre à l'utilisateur de modifier le vectoriel et le transformer à sa guise.

Une troisième et dernière partie liée à la classification des pierres, nous permettra, premièrement, d'établir quelles sont les catégories de pierres que le CAPRA désire caractériser. Avec l'aide du CAPRA, nous établirons les caractéristiques de ces pierres que nous traduirons en mathématiques. Deuxièmement, ce chapitre se terminera par l'implémentation informatique des calculs liés à la classification automatique des pierres puis à la retouche des résultats produits. Mais avant cette retouche faite par l'utilisateur, les résultats produits automatiquement par cette classification seront étudiés.

2. Présentation du contexte

La chaîne de traitement et d'acquisition des données que nous souhaitons développer s'articulera de la façon suivante : Tout d'abord, l'utilisateur acquiert des mesures de la façade à traiter en la prenant en photographie (le passage en orthophoto est facultatif) puis il transmet cette donnée, par le biais d'une interface graphique, à un outil qui vectorise de manière automatique l'image et fait ressortir les contours des pierres, briques, etc. Cet outil est programmé en Python (langage de programmation répandu dans le monde professionnel et étudié à l'ESGT). Cette segmentation alors produite, loin d'être parfaite, est retouchée par l'utilisateur pour devenir l'équivalent d'une vérité terrain. La segmentation alors terminée, l'outil doit, en fonction d'indices prédéfinis (de couleur, forme, texture, taille prédéfinies), caractériser les pierres délimitées de manière à ce qu'elles soient associées à une catégorie de pierre. Ces catégories sont déterminées en collaboration avec le CAPRA. Pour terminer, l'utilisateur retouche les erreurs de cette qualification. Cette classification se déroule sous ArcMap (logiciel du pack ArcGIS).

Pour une meilleure compréhension du mode opératoire décrit ci-dessus, nous le résumons dans le schéma suivant :



Le CAPRA utilise des orthophotographies au format TIFF (figure 1). Les archéologues produisent à partir de ces dernières des fichiers vectoriels représentant les contours des objets concernées.

Figure 1: Image d'une rue pavée et du traitement vectoriel produit par le CAPRA manuellement

La figure 1 est une prise de vue de pavés de sol. Le CAPRA souhaitait aussi connaître les capacités de l'outil à déterminer les contours de pavés de sol. Nous n'avons donc pas de joints entre les pierres. Néanmoins, pavés de sol ou pierre de mur sont des cas similaires vis-à-vis du traitement.

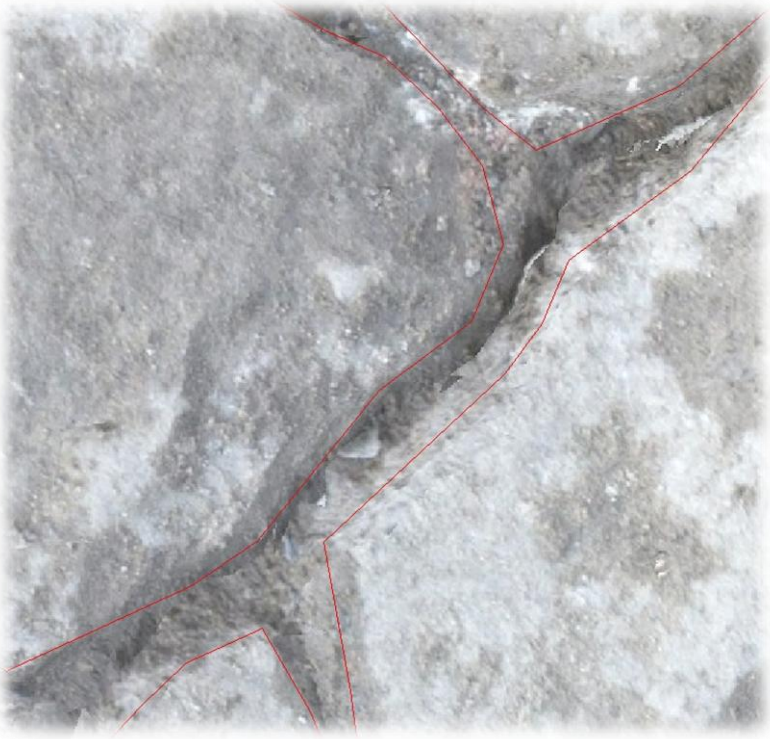


Figure 2: Zoom de la figure 1

Les contours sont bien représentatifs des éléments mais ils ne suivent pas au pixel près les contours véritables des pierres (figure 2). On ne peut pas toujours établir exactement les contours des objets que ce soit à l'œil nu ou par informatique.

Dans notre cas, nous avons établi avec le CAPRA de faire fonctionner en priorité un outil avec des orthophotographies de la cathédrale du Mans puisque le CAPRA coordonne un projet d'études et de relevé de cet édifice. Nous avons pour cela choisi une image test (figure 3).

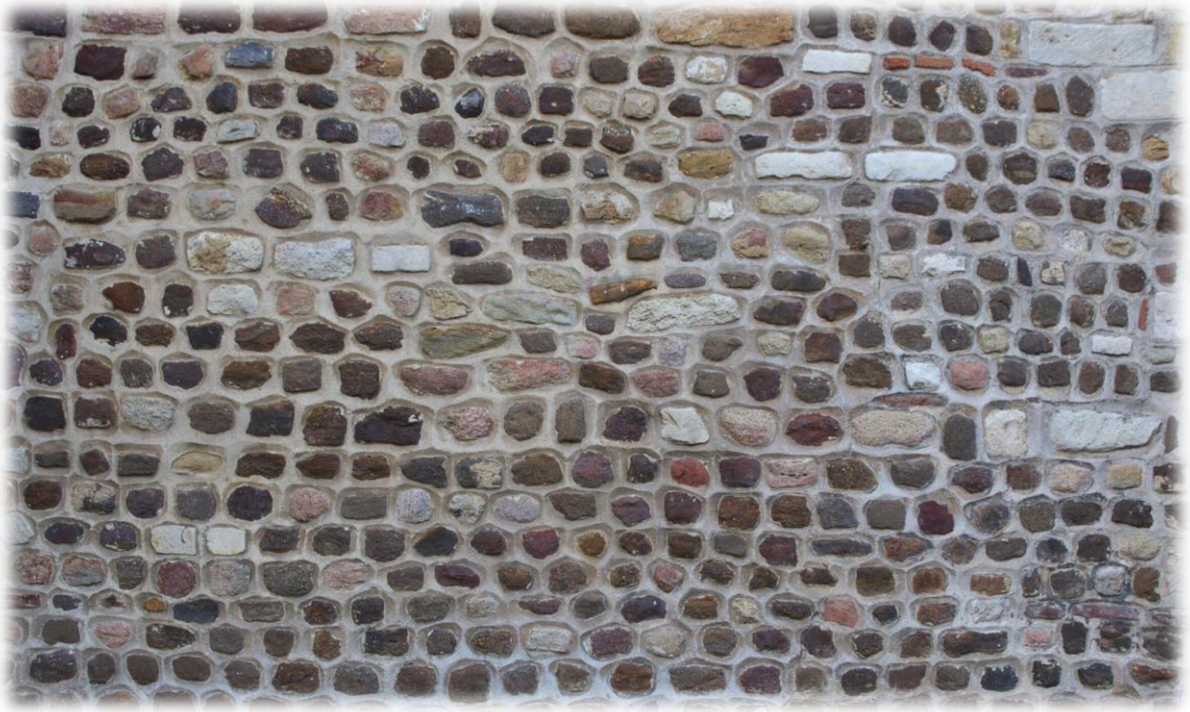


Figure 3: Photographie d'un petit appareil de moellons assisé et régulier (mur de la cathédrale du MANS)

Cette image sera la base de tous nos tests et donc de toutes nos conclusions. Nous l'avons choisie car les pierres et les joints sont bien différenciés tant au niveau des couleurs que dans la propreté de réalisation des joints. De plus, il y a une grande diversité de nature et de taille des pierres. En utilisant cette image, on pourra donc travailler sur une manne d'informations à la fois simple et pourtant diversifiée.

Nous avions émis l'idée de produire un outil qui, après traitement et rectification des contours de pierre, nous classifie ces pierres suivant la catégorie à laquelle elles appartiennent. Pour cela, nous avons établi avec le CAPRA une liste de 7 catégories de pierre. Nous avons limité les catégories à ce que le CAPRA peut appréhender dans ses activités locales :

- La brique (1),
- Le calcaire de Bernay (2),
- Le granit (3),
- Le grès cénomanien (4),
- Le grès éocène (5),
- Le grès roussard (6),
- Le tuffeau (7).

Les premières caractéristiques établies brièvement par le CAPRA sont répertoriées dans l'annexe n°2. Néanmoins voici une illustration pour chacune des 7 catégories (figure 4) :

Figure 4 : Images des 7 catégories de pierres

3. Segmentation

Le traitement d'une image numérique et plus particulièrement la segmentation sont des domaines déjà apprivoisés. Il existe de nombreux algorithmes déjà développés. Pour notre part, le travail durant 5 mois, nous n'avons pas souhaité nous pencher sur le développement de notre propre algorithme pour étudier l'ensemble de l'opération. Nous avons donc réutilisé des sources existantes. Tout d'abord, nous avons étudié l'Orfeo ToolBox (OTB) couplé au logiciel Monteverdi, bibliothèque de traitement d'image relativement connue. Puis nous avons exploré la bibliothèque Scikit, cette dernière étant développée en Python. Ces deux approches ont été choisies car elles sont libres. Ces algorithmes sont brièvement présentés et les paramètres de réglage dont ils dépendent sont énumérés dans l'annexe n°1.

Sur chacun des résultats qui sont présentés dans ce mémoire, les contours délimitent les pierres. A l'intérieur de ces contours, les pixels sont à 1 ou 255 et en dehors, ils sont à 0 (ou NULL).

3.1. Orfeo ToolBox/Monteverdi

3.1.1. Présentation de l'OTB

Développée par le CNES (Centre National d'Études Spatiales), pour le traitement de grandes images spatiales, l'Orfeo ToolBox (OTB) est une bibliothèque (« Library ») d'algorithmes (CNES, 2014). Elle comporte divers algorithmes de traitement d'image fondés sur une autre bibliothèque : ITK (Insight ToolKit : bibliothèque de traitement des images médicales). L'OTB est distribuée sous la licence CeCILL[2] : elle est donc open Source. Les développeurs souhaitent encourager la recherche et les contributions des utilisateurs autour de ce projet.

Plus particulièrement, l'OTB compte 5 algorithmes de segmentation différents que nous verrons dans la partie suivante :

- Mean-Shift
- Edison Mean-Shift
- Connected Components
- Watershed
- Morphological profiles based segmentation

[2] CEa Cnrs Inria Logiciel Libre : Le but de cette licence est d'accorder aux utilisateurs le droit de modifier et de redistribuer le logiciel régi par cette licence dans le cadre d'un modèle de diffusion en logiciel libre (licence GNU GPL). Elle garantie le respect du droit français aux créateurs et utilisateurs de logiciel libre.

Le logiciel Monteverdi développé par le CNES, permet d'utiliser, à travers une interface, les fonctionnalités énoncées précédemment. Nous l'utiliserons aussi pour tester les algorithmes.

3.1.2. Segmentation *Mean-Shift*

L'estimateur de la densité du gradient, le *Mean-Shift* utilise différentes gammes de valeurs, répertoriées dans deux domaines, pour, entre autres, produire une segmentation d'image. L'espace du maillage est connu comme le domaine spatial (« spatial domain ») tandis que les informations des pixels (niveau de gris, couleurs...) sont représentées dans ce l'on appelle le « range domain » (domaine des attributs). Les pixels comportent donc 5 informations : X, Y et R, G, B[3] ou H, S, V[4]. Le principe de cet algorithme est de rechercher les « modes » ou *maxima* locaux de densité d'une répartition donnée.

La segmentation d'une image par cette méthode se déroule en deux étapes :

- On associe (par convergence) tout d'abord à chaque pixel de l'image, le mode le plus proche,
- Puis, on fusionne les pixels d'un même mode pour former une région.

Durant le processus itératif, l'algorithme recherche dans une aire donnée (en fonction du rayon spatial) le centre des masses (= *maxima* local) de densité. La distance entre le point initial et le centre des masses de la zone forme le vecteur Mean-Shift (figure 5).

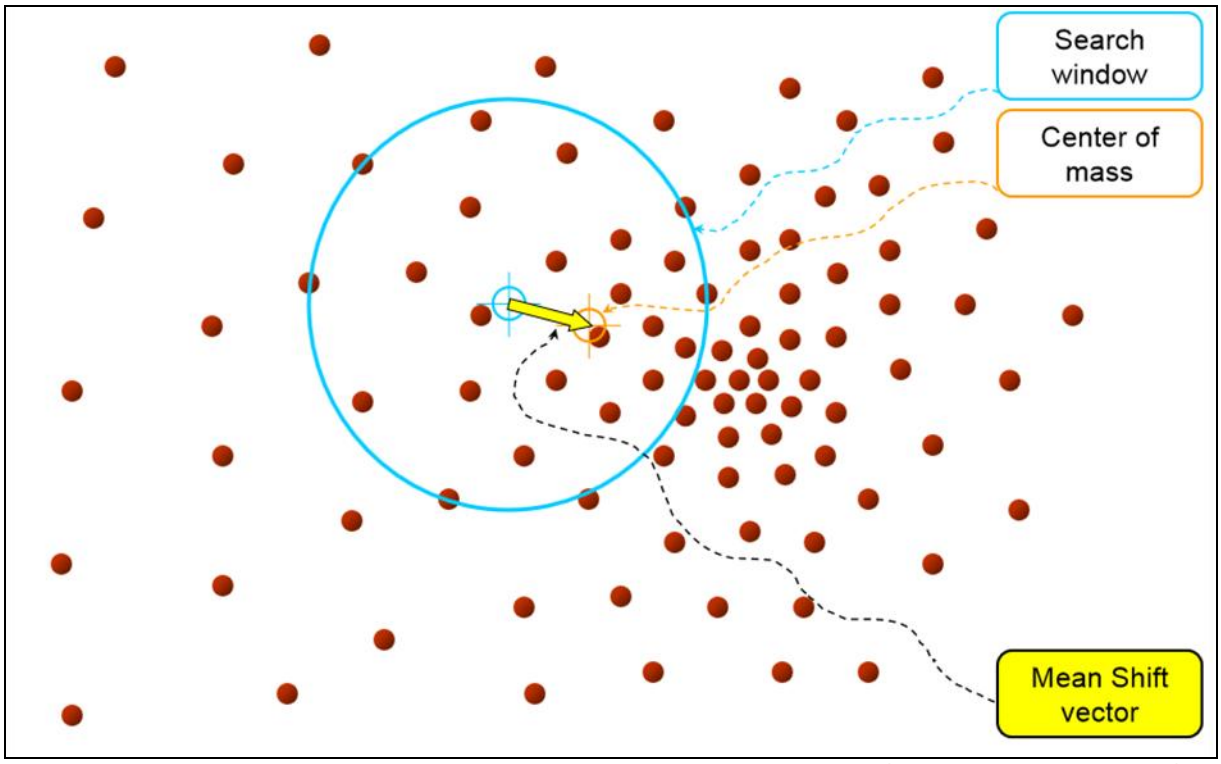


Figure 5: Représentation de la recherche du centre des masses dans une aire donnée (Y. Ukrainitz et B. Sarel, 2003)

[3] RGB est un mode de représentation des images couleur, R = rouge, G = vert et B = bleu.
[4] HSV est un autre système de représentation des images couleur, H = teinte, S = saturation et V = luminosité.

Pour calculer le centre des masses, on utilise :

$$Y_{k+1} = \frac{1}{n_k} \sum_{x_i \in S_h(y_k)} x_i$$

Où :

Y_k est le centre de la fenêtre à l'étape k avec k = 1,2, …,

S_h est la fenêtre centré en Y_k,

n_k est le nombre de points appartenant à la fenêtre de l'étape k,

x_i est un point de l'image.

Une fois chaque pixel associé à un centre des masses, l'algorithme forme des régions regroupant toutes les trajectoires menant à un même mode, pour chacun d'entre eux (figure 6).

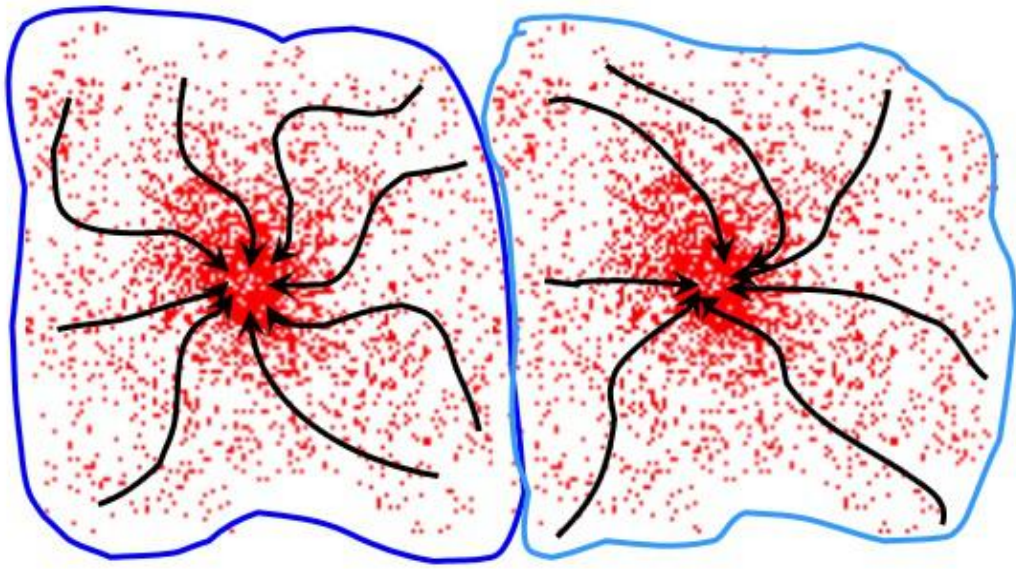


Figure 6: Représentation de l'agrégation des points ayant le même centre des masses en une région (Y. Ukrainitz et B . Sarel, 2003)

Quant à la méthode nommée *Edison Mean-Shift*, elle est fondée sur les régions de confiance et la segmentation *Mean-Shift* (CNES, 2014). Elle comporte deux étapes principales :

- Le filtrage : application de l'algorithme *Mean-shift* sur chaque pixel de l'image jusqu'à convergence.

- Segmentation et labellisation : une image est produite à partir du fichier de sortie formée par agrégation des pixels. Puis une étape de fusion est effectuée (les régions adjacentes avec une différence spectrale inférieure à la gamme spectrale précisée sont fusionnées). Finalement, les régions trop petites sont agrégées avec la région spectrale adjacente la plus proche.

3.1.3. Segmentation *Connected Components*

L'objectif de cet algorithme est d'effectuer la segmentation par composantes connexes avec un critère de segmentation défini par l'utilisateur (CNES, 2013). Ce critère est une équation appliquée à chaque pixel de l'image.

Après que la segmentation ait été appliquée, l'algorithme labellise les objets : un niveau de gris différent est attribué à chaque composante.

3.1.4. Segmentation *Watershed*

La segmentation par ligne de partage des eaux considère une image de niveau de gris comme un relief topographique dont on simule l'inondation (Wikipedia Foundation, 2013). On calcule le gradient de l'image. Les hautes intensités (proches du blanc) forment les sommets des collines et les basses intensités (proches du noir) forment les vallées. Lorsque le processus commence, chaque vallée est remplie par de l'eau. Chaque vallée a sa propre couleur. Lorsque deux bassins de couleurs différentes se rencontrent, on crée une ligne de partage qui empêche leur fusion. Lorsque tout le relief est sous l'eau, le processus s'arrête. Les lignes de partage des eaux forment alors notre résultat (OpenCV dev Team, 2014).

3.1.5. Segmentation *Morphological profiles based*

Cette méthode de segmentation d'image utilisant la morphologie mathématique est fondée sur deux outils principaux : la transformation en bassins versants de l'image et la transformation homotopique qui résout le problème de la sur-segmentation et introduit la notion de marqueurs des objets à segmenter. Les marqueurs des objets à segmenter permettent, avant l'application du *Watershed*, de savoir quelles sont les zones d'intérêt et quelles sont les zones d'arrière-plan. Dans l'utilisation basique de la segmentation par ligne de partage des eaux, la distinction de ces zones n'est pas faite, ainsi on se retrouve avec des lignes détectées qui ne devraient pas l'être. Pour y remédier, on applique une modification homotopique qui change le *minima* de l'image par un autre, l'équivalent des marqueurs d'arrière-plan (il enlève du processus les zones qui ont été considérées comme arrière-plan par l'utilisateur) (Meyer, 1992).

3.2. Python : Scikit-Image

3.2.1. Présentation de Scikit-Image

Scikit-Image est une bibliothèque d'algorithmes pour le traitement d'image dans le langage de programmation Python. Elle est *open-source*.

Elle propose principalement les 5 algorithmes suivants :

- *Edge Based Segmentation,*
- *Region based Segmentation,*
- *Random walker,*
- *Felzenszwalb,*

3.2.2. Segmentation *Edge Based*

Un moyen simple pour segmenter une image est de choisir un seuil sur la base de l'histogramme des niveaux de gris de cette image. Malheureusement, appliquer un seuil sur une image peut produire un résultat dans lequel il manque de nombreuses zones d'intérêt (valeurs en-dessous du seuil supprimées).

Pour ce faire, nous obtenons tout d'abord les contours en utilisant le détecteur *Canny edge-detector*.

L'algorithme fonctionne de la manière suivante :

1- Filtrage gaussien (Passe-Bas) pour limiter le bruit de l'image

2- Calcul des gradients des niveaux de gris de l'image par filtrage de Canny (Passe-Haut). Pour cela, il calcule le gradient vertical puis le gradient horizontal et termine en obtenant de ces deux derniers la norme des gradients.

3- Seuillage par Hysteresis avec un seuil haut et un seuil bas pour détecter les points de contour.

4- Squelettisation des contours. C'est-à-dire amincissement des contours à 1 pixel. Dans le cas de contours larges, il garde le pixel qui à la plus forte norme dans la direction du gradient.

5- Ces contours (formant des régions) sont ensuite remplis en utilisant une morphologie mathématique, ceci ne fonctionne que si les contours sont fermés.

6- Élimination des objets de petite taille par la mise en place d'une taille minimale.

3.2.3. Segmentation *Region Based (~Watershed)*

Tout d'abord, nous trouvons une carte d'élévation en utilisant le gradient de Sobel de l'image (nous pouvons appliquer un masque pour limiter la zone de traitement). Puis, nous trouvons des marqueurs de l'arrière-plan et des régions d'intérêt en nous fondant sur les valeurs extrêmes des niveaux de gris de l'image.

Pour terminer, nous utilisons la transformée en bassins versants pour « remplir » les zones de la carte d'élévation à partir des marqueurs déterminés ci-dessus. La transformée en bassins versants est expliquées plus profondément dans la partie 3.1.4..

3.2.4. Segmentation *Random Walker*

De la même manière que le précédent algorithme, le *Random walker* détermine la segmentation d'une image à partir d'un ensemble de marqueurs d'étiquetage (de régions d'intérêt et d'arrière-plan) (Scikit-image development team, 2014). Une équation de diffusion est résolue avec les traceurs initialisés à la position des marqueurs. Le coefficient de diffusion est plus grand si les pixels voisins ont des valeurs similaires, de sorte que la diffusion soit difficile à travers des gradients élevés. Chaque pixel reçoit l'étiquette du marqueur connu qui a la plus forte probabilité d'être atteinte au cours de ce processus de diffusion.

3.2.5 Segmentation *Felzenszwalb*

Cet algorithme de segmentation d'image a un paramètre d'échelle qui influe sur la taille des segments (Scikit-image development team, 2014). Une image RGB verra ses 3 canaux traités individuellement. Pour former le résultat final, l'algorithme ne gardera que les segments appartenant aux 3 canaux (intersection des 3 canaux). La taille réelle et le nombre de segments peuvent varier considérablement en fonction du contraste local.

3.3. Expérimentations

Les bibliothèques dont nous avons parlé précédemment nous offrent un panel d'une dizaine d'algorithmes, mais comment choisir lequel ou lesquels à implémenter dans notre outil ?

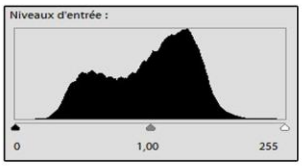
Pour effectuer ce choix, nous avons mis en place une batterie de tests.

3.3.1 Prétraitement

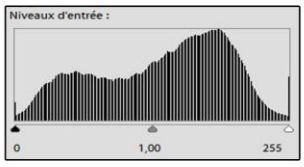
Lors de cette étude nous procédons toujours avec l'image figure 3.

*** Étalonnage**

L'étalonnage permet de répartir l'histogramme des niveaux de gris de l'image entre 0 et 255 (figure 7) alors que certaines valeurs proches de 0 et de 255 n'étaient presque pas présentes dans l'histogramme.



Histogramme de l'image test vierge de tout traitement



Histogramme de l'image test après étalonnage

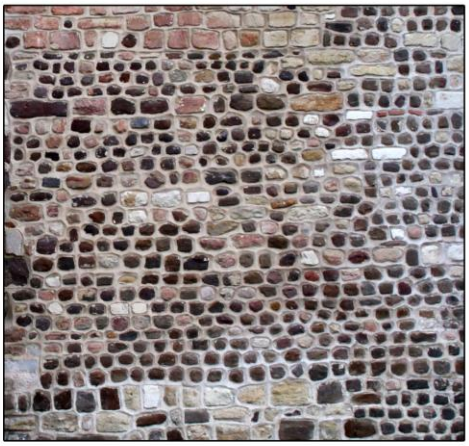


Image test après étalonnage

Figure 7 : Influence du rééchantillonnage de l'histogramme

Étalonner les niveaux de gris proches de 255 (proche du blanc) implique un éclaircissement des pixels proches de cette valeur puisque (par exemple) ceux aux alentours de 200-210 valent à présent 240-255. De même, ceux proches du noir (0), se sont assombris pour se rapprocher d'un noir plus pur. De manière générale, le contraste entre pixels sombres et clairs s'est accentué.

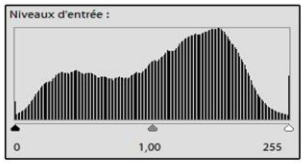
Attention : Réduire trop l'étalement pour accentuer les couleurs proches du blanc, n'est pas une bonne idée. Cela va assombrir le reste de l'image aussi (ex : joints). L'algorithme va donc détecter beaucoup plus de pierres, mais aussi beaucoup plus d'ombres et contours parasites. On va donc devoir retoucher aux paramètres afin de supprimer un maximum ces nuisances : on retournera alors sur un résultat similaire à celui de départ. Cet étalement restreint nous aura demandé beaucoup de temps pour régler de nouveau les paramètres.

*** Luminosité/contraste**

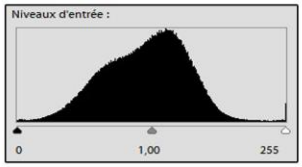
Faire varier ces paramètres semble être une bonne idée. Cependant, cela ne règle pas le problème : les pierres les plus claires restent toujours proches de la colorimétrie des joints, si l'on veut supprimer l'un, l'autre disparaît avec. L'algorithme ne repère que les fortes zones d'ombres au contact de ces pierres claires puisqu'il se base sur l'échelle de niveaux de gris.

*** Contraste *(bis)***

Dans ce cas, nous avons joué sur le contraste d'une autre manière : les pixels foncés ont été éclaircis, ceux plutôt clairs ont été foncés (figure 8). Les pierres de couleurs claires semblent se différencier des joints claires.



Histogramme image test après étalonnage



Histogramme image après application de ce traitement

Image résultante

Figure 8 : Étalonnage de l'histogramme

*** Saturation**

Augmenter la saturation de l'image permet une détection plus importante des pierres mais apporte quelques défauts à celles déjà détectées (figure 9).

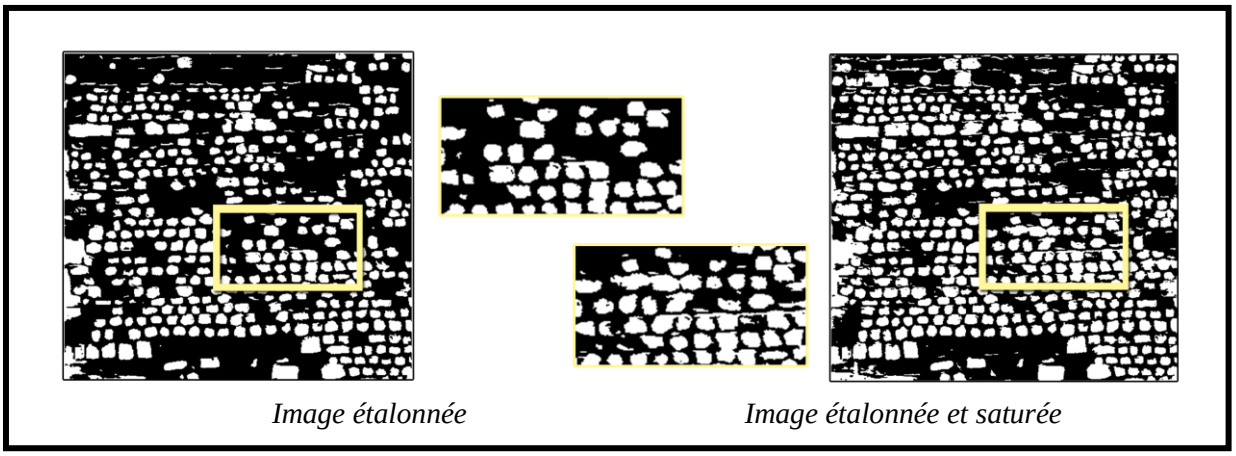


Figure 9 : influence de la saturation sur le résultat après la segmentation

*** Autres ...**

D'autres prétraitements comme la variation de couleur, teinte, accentuation de l'image furent aussi expérimentés… Tous les résultats mènent aux mêmes conclusions :

Ces divers prétraitements ont un impact sur nos résultats. A paramètres égaux, nous avons pu voir que les prétraitements engendrent des résultats complètement différents. Néanmoins, le rendu est toujours le même : nous obtenons soit une détection riche mais fausse à plusieurs endroits, soit une détection plus pauvre mais dont les pierres détectées sont correctes. Il apparaît aussi que ce rapport entre « nombre de pierres détectées » et « qualité de détection » peut être ajusté via les paramètres des méthodes de segmentation : nous pouvons obtenir les mêmes résultats, obtenus avec prétraitement, en modifiant seulement les paramètres des algorithmes de traitement d'image.

3.3.2. Protocole d'expérimentation

Tout d'abord, nous avons étudié chacun des algorithmes indépendamment en évaluant l'influence de leurs paramètres d'entrée sur le résultat de sortie. Pour cela, nous faisons varier un seul paramètre. On qualifie les résultats produits de manière grossière (à l'œil nu) en les comparants les uns aux autres selon les points suivants :

- le nombre de pierres détectées dans l'image,
- la bonne correspondance des contours détectés avec la réalité,
- le nombre d'objets détectés n'étant pas des pierres.

Cela nous a permis, à la fois de comprendre l'influence des paramètres sur le résultat, mais aussi d'obtenir un bon réglage des paramètres optimum pour les images test.

A la suite de cette analyse, nous avons retenu un algorithme. Nous avons mis en place une seconde phase d'expérimentation plus précise, avec des calculs permettant de quantifier nos résultats produits par l'algorithme sélectionné.

Pour notre image « test », nous possédons la vérité terrain approuvée par le CAPRA, ainsi que le résultat retourné par notre algorithme de traitement d'image (segmentation).

Nous avons alors mis en place des calculs d'indices sous ArcGIS :

- Premièrement, on compte le nombre de pixels appartenant à la fois à une pierre de la vérité terrain et à la fois à une pierre de notre résultat automatique. On divise ce chiffre par le nombre de pixels faisant partie, uniquement, des pierres de la vérité terrain. On obtient alors une probabilité de détection.
  - $Indice\ 1 = \frac{a}{b} * 100$

  Où a : le nombre de pixel à 1 à la fois dans la vérité terrain et dans le résultat automatique

  b : le nombre de pixel à 1 sur la vérité terrain

- Le dernier indice mis en place est celui de la probabilité de fausse alarme. Nous avons compté le nombre de pixels à 0 sur la vérité terrain mais à 1 (ou 255) sur le résultat automatique puis nous l'avons divisé par le nombre total de pixels à 0 sur la vérité terrain. Cet indice est, comme le premier, calculé sur toute l'image.
  - $Indice\ 2 = \frac{a}{b} * 100$

  Où a : le nombre de pixel à 0 sur la vérité terrain mais à 1 sur le résultat automatique

  b : le nombre de pixel à 0 sur la vérité terrain

- Le premier résultat étant généralisé à notre raster, nous avons aussi mis en place un calcul pour obtenir le résultat associé à chaque composante connexe (chaque pierre indépendamment des autres). Ce dernier, spécifique à chaque pierre, nous permet de savoir si chacune d'elles est ou non détectée. Pour cela nous avons mis en place un seuil de 90 % au-dessus duquel nous avons considéré qu'une pierre était détectée. A 0 %, nous avons conclu que les pierres n'étaient pas détectées.

  $Pour\ chaque\ pierre$:

  - $Si\ Indice\ 1\ > 90, pierre\ détectée$
  - $Si\ indice\ 1 = 0, pierre\ non - détectée$

Attention : Cette qualification possède des limites puisqu'une pierre étant classée comme détectée car au moins 90 % de ses pixels sont à la fois dans la vérité et dans le résultat automatique n'est pas forcément bien détectée (si les contours sont plus larges que la pierre à cause de l'ombre, d'un gradient faible, d'une autre pierre très proche...etc). Comme nous l'avons dit au paragraphe p.8 « 2. Présentation du contexte », certains contours de pierre ne sont pas faciles à replacer.

Ce projet ayant pour but de réduire le temps de traitement des archéologues, nous avons cherché à privilégier lors de ces expérimentations des résultats dont les pierres détectées sont peu nombreuses mais relativement justes (image gauche, figure 10) plutôt que des résultats où une majorité de pierres seront détectées mais mal et dont l'utilisateur devra retoucher ou supprimer les contours (image droite, figure 10).

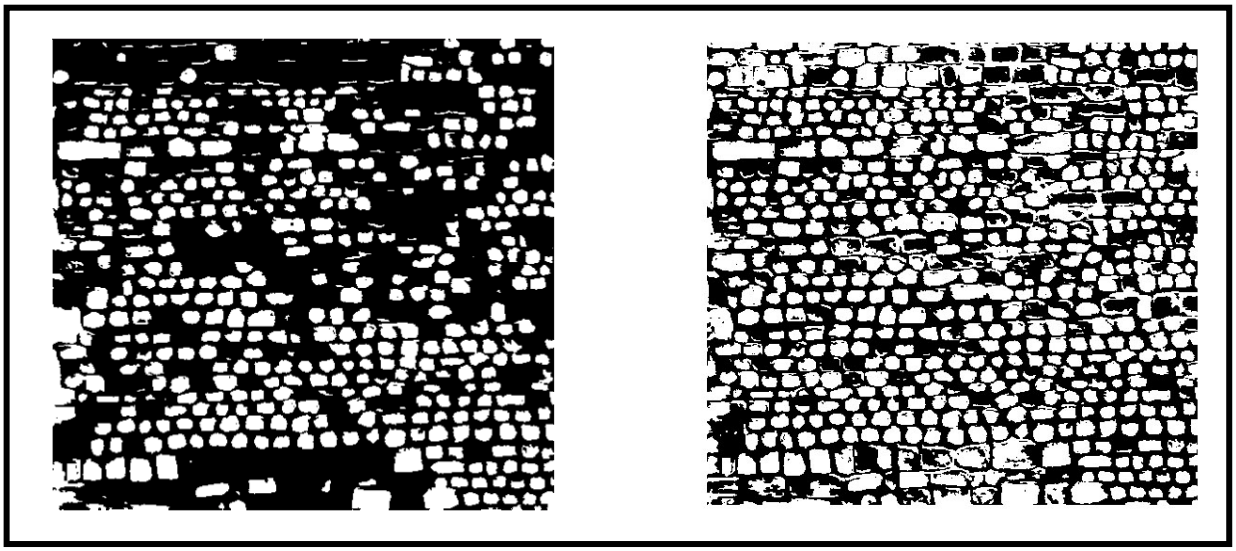


Figure 10 : Résultats obtenus après la même segmentation pour différents paramètres

3.3.3. OTB

Intéressons nous tout d'abord à la segmentation *Mean-Shift*.

Lorsque l'on fait varier le paramètre range radius, on remarque que plus il augmente et plus le traitement de l'image prendra de temps pour s'effectuer.

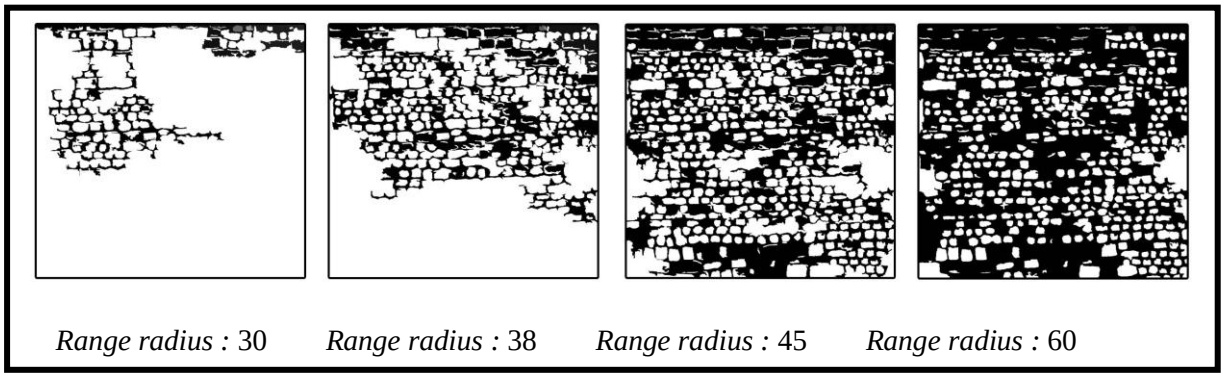


Figure 11 : Impact du paramètre *range radius* de l'algorithme *Mean Shift*

Si ce dernier est trop grand, le résultat va perdre en qualité (figure 11). Certains contours intéressants, vont simplement disparaître comme par exemple à l'endroit de l'encadré circulaire (figure 12). Ainsi l'augmentation de ce paramètre suit le schéma « de nombreuses pierre détectées mais peu le sont biens - > peu de pierres détectées mais beaucoup le sont biens ».

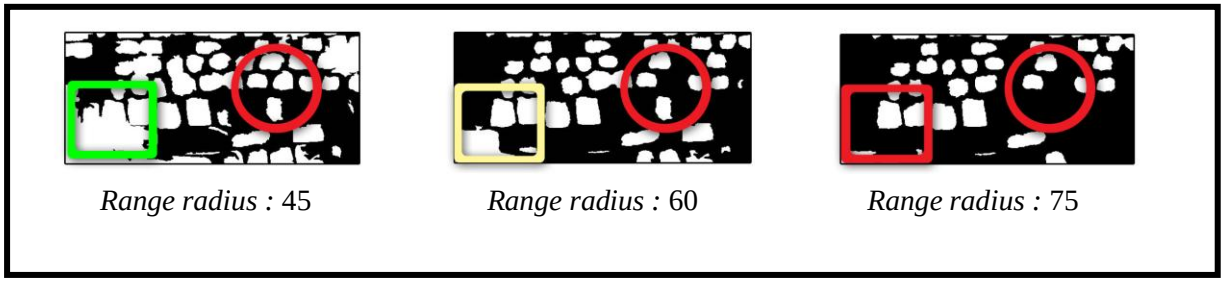


Figure 12 : Impact du paramètre *range radius* (Zoom figure 8)

Le traitement n'a pas le même impact à chaque endroit de l'image. Il faut donc trouver le résultat qui nous convienne le mieux puis conserver le *range radius* correspondant.

On fixe à présent le range radius à 45 puisqu'il paraît nous retourner un des meilleurs résultats. Nous allons étudier l'impact du paramètre *spatial radius* (figure 13).

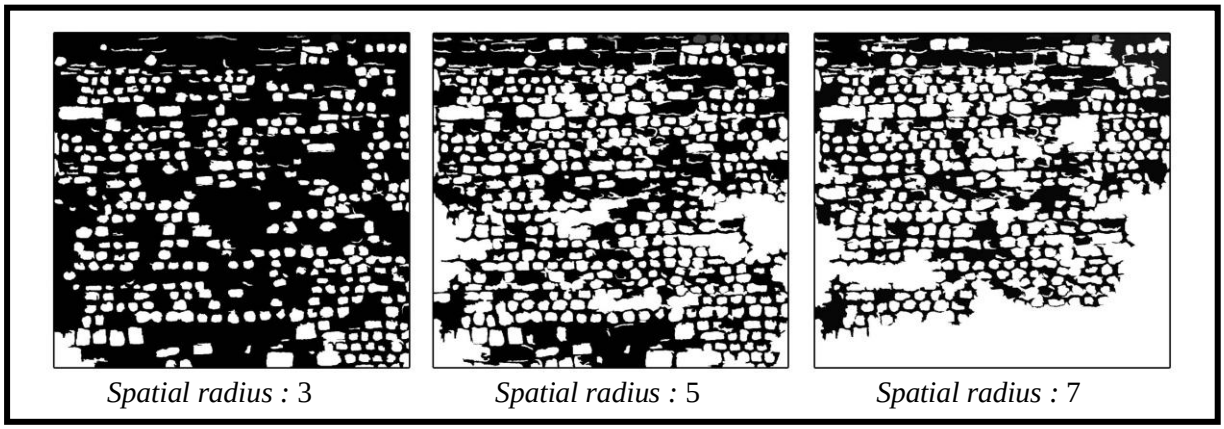


Figure 13 : Impact du paramètre *spatial radius* de l'algorithme *Mean Shift*

Plus ce rayon est grand et plus nous détectons de contours, néanmoins s'il est trop grand, certaines pierres ne sont plus différenciées. Il faut ainsi trouver le juste compromis entre « Peu de pierres détectées mais bien délimitées » et « nombreuses pierres détectées mais mal délimitées ». Les deux paramètres *spatial radius* et *range radius* ont des effets complémentaires. Si l'on augmente radicalement l'un, il faut faire de même avec l'autre pour compenser (figure 14).

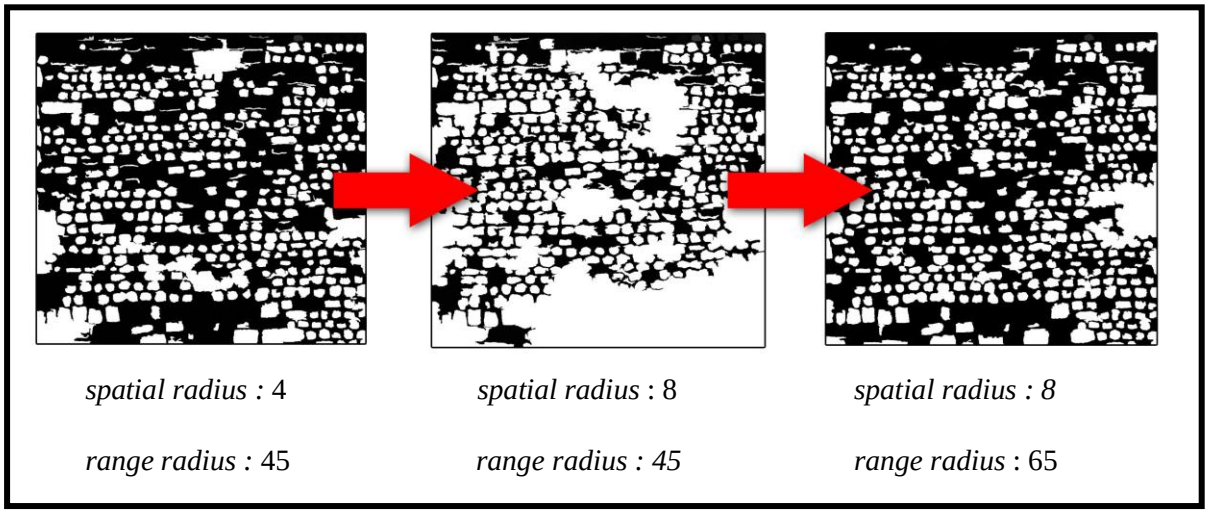


Figure 14 : Représentation de la compensation entre les paramètres *spatial radius* et *range radius*

Concernant le paramètre du seuil de convergence, si nous augmentons ce dernier, nous obtenons une meilleure délimitation des pierres (encadrés verts, figure 15), mais nous perdons aussi certaines informations (cercles rouges, figure 15).

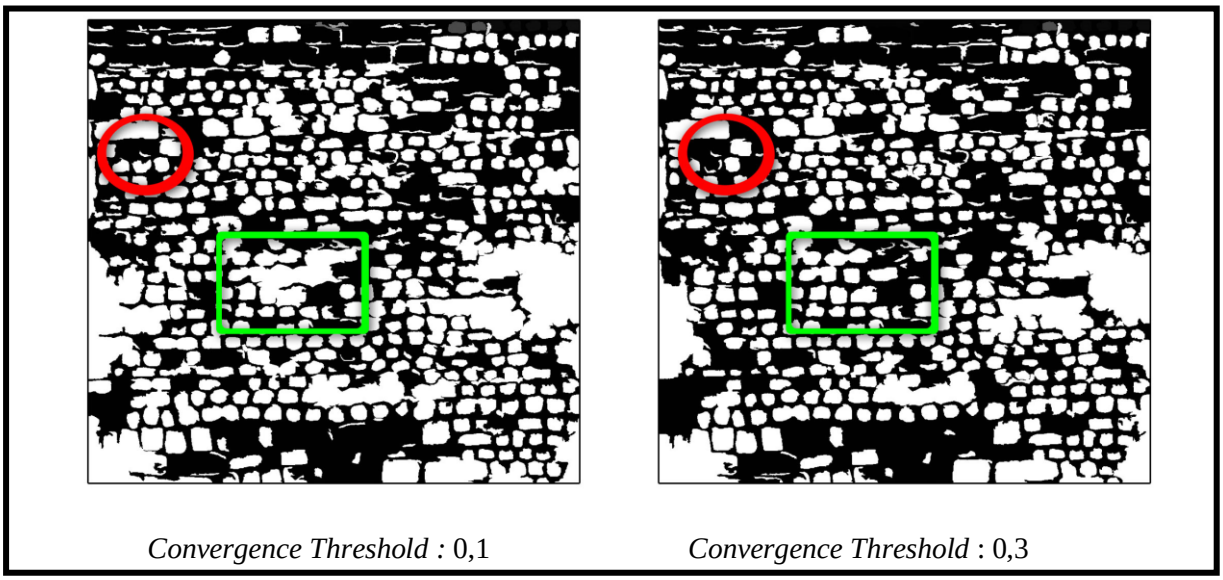


Figure 15 : Impact du paramètre *convergence threshold* de l'algorithme *Mean Shift*

Quant au paramètre *minimum region size,* il permet de réduire le nombre d'éléments détectés qui ne sont pas des pierres (figure 16).

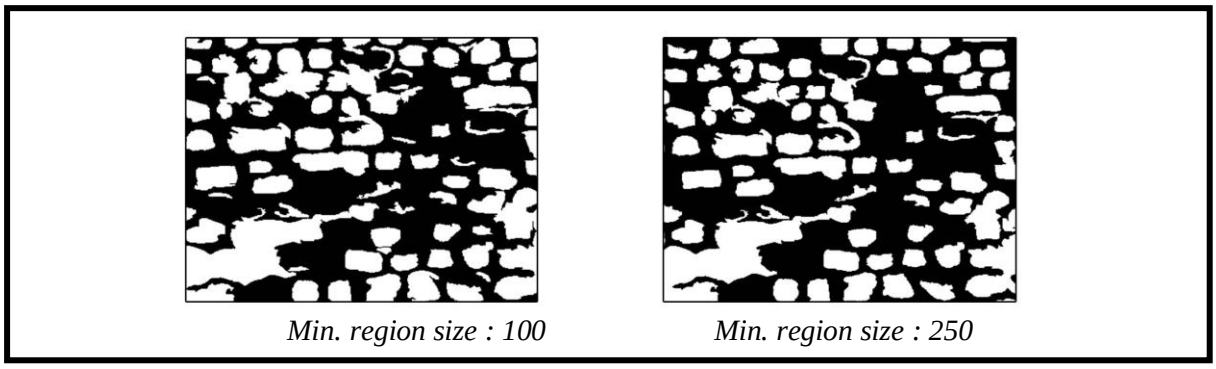


Figure 16 : Effet du paramètre *minimum region size* sur le résultat obtenu avec l'algorithme *Mean-Shift*

Penchons-nous à présent sur le cas de l'*Edison Mean-Shift*.

Nous notons que les mêmes valeurs des paramètres précédents dans ce traitement ne donnent pas le même résultat qu'avec l'algorithme *Mean-shift* (figure 17).

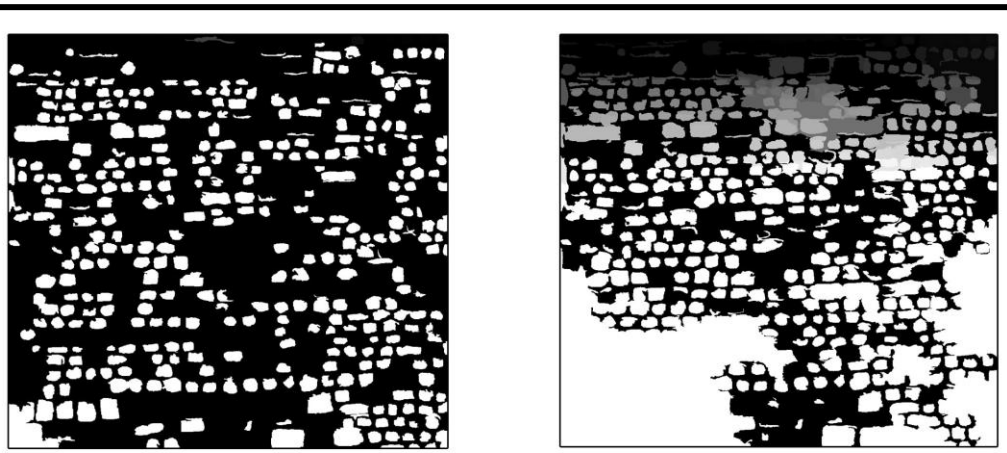


Figure 17 : Comparaison des deux algorithmes *Mean Shift* et *Edison Mean Shift* en utilisant les mêmes valeurs de paramètres

Le paramètre *spatial radius* semble avoir beaucoup plus d'impact sur le résultat, et le simple fait de passer sa valeur de 3 à 4 (par exemple) change fortement le résultat (figure 18).

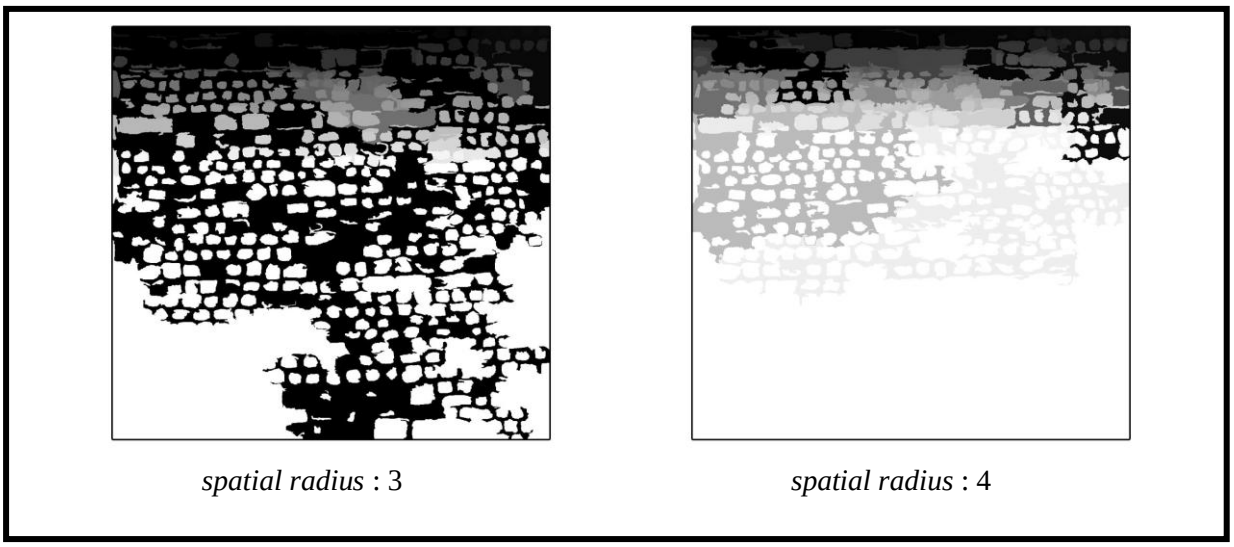


Figure 18 : Impact du paramètre *spatial radius* de l'algorithme *Edison Mean Shift*

Pour compenser cela, il faut donc d'autant plus augmenter le paramètre *range radius*. Ainsi, pour un spatial radius de 4 dans l'algorithme *Mean-shift*, une bonne valeur (dans le cas de notre image « test ») pour le *range radius* serait d'environ 40-45, alors que dans l'algorithme *Edison Mean-Shift*, elle est d'au moins 60-65.

Notez que pour ces deux premiers algorithmes, l'augmentation du paramètre *spatial radius* engendre une augmentation notable du temps de traitement.

Continuons sur l'algorithme relativement connu qu'est le *Watershed*.

Premièrement, concernant le seuil (*Depth Threshold*), si celui-ci est trop haut, de nombreuses pierres « trop claires » ne seront pas prises en compte, et inversement si le seuil est trop bas, les pierres seront regroupées en une même région (figure 19).

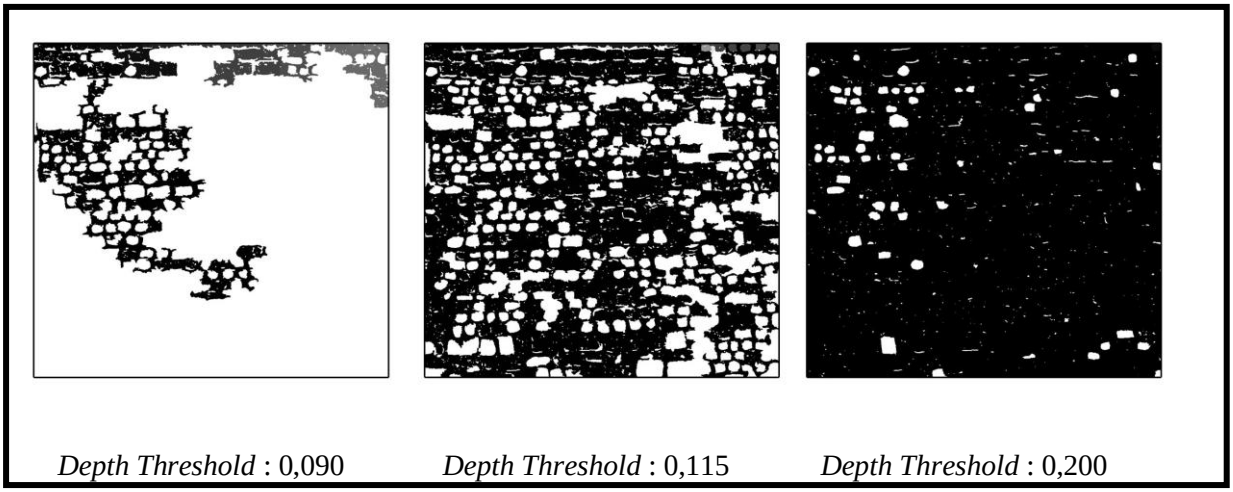


Figure 19 : Impact du paramètre *depth threshold* de l'algorithme *Watershed*

De plus, nous pouvons constater que cet algorithme génère de nombreuses régions « parasites ». Il faudrait alors y ajouter un autre paramètre de *minimum region size* pour pouvoir supprimer la plupart de ces dernières.

Le paramètre *flood level* (« niveau d'inondation ») influe beaucoup moins que le paramètre *depth threshold* sur les résultats. Pour un niveau d'inondation faible (proche de 0), les résultats obtenus restent convenables. Cependant, un niveau trop haut (supérieur à l' écart maximum d'altitudes), ne permet pas de détecter les pierres (figure 20).

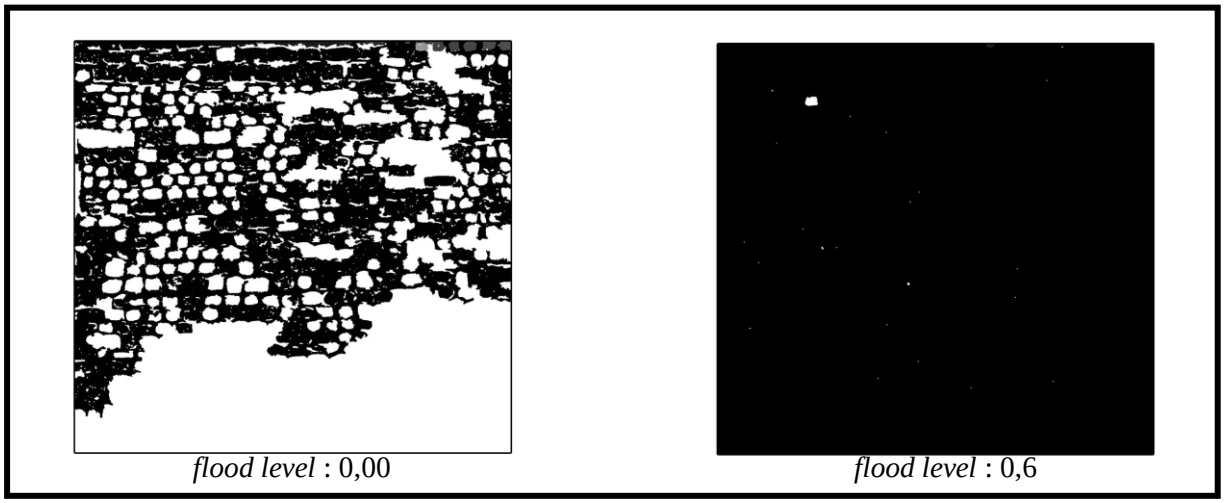


Figure 20 : Impact du paramètre *flood level* à travers l'application de l'algorithme *Watershed*

Pour terminer cette première partie, intéressons nous à l'algorithme *Morphological Profiles Based Segmentation*.

Les tests concernant cette méthode sont brefs : on obtient des résultats prometteurs mais les temps de traitement sont conséquents et la mémoire vive utilisée est trop importante pour un ordinateur basique.

L'augmentation du paramètre *initial radius* permet de détecter de nombreuses pierres mais un bon tiers l'est mal. Pour améliorer le résultat, il faut augmenter le *radius step*. (figure 21).

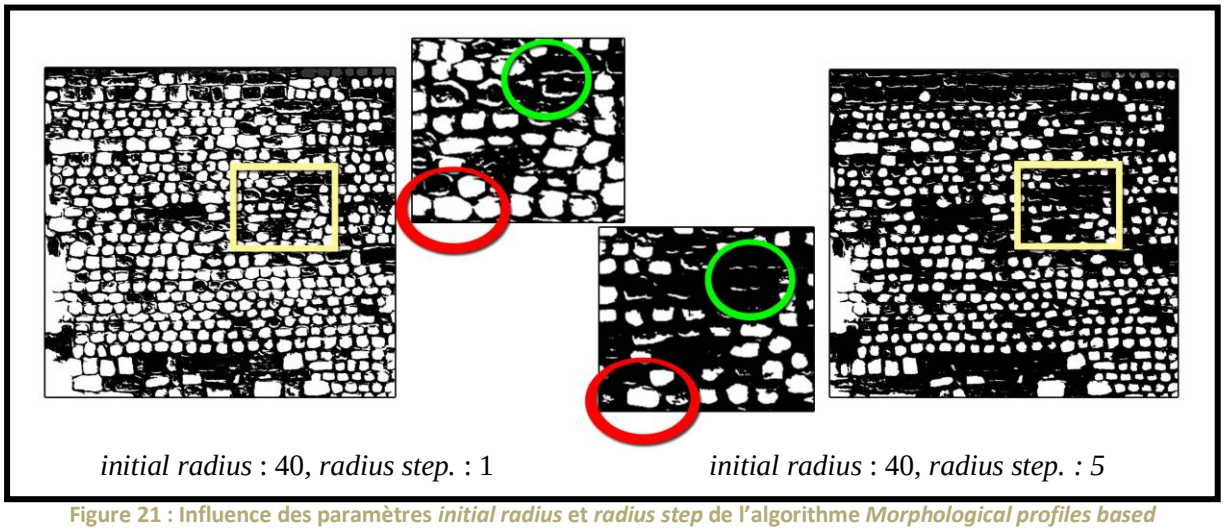


Figure 21 : Influence des paramètres *initial radius* et *radius step* de l'algorithme *Morphological profiles based segmentation*

Ce dernier paramètre engendre un phénomène d'érosion : il va permettre d'enlever/de réduire les pierres mal détectées (cadre vert, figure 21) mais il va aussi réduire la taille de certaines bien détectées (cercle rouge, figure 21).

Que pouvons-nous retenir de tous ces tests ? Intéressons-nous à présent aux possibilités qu'offrent ces algorithmes mais aussi aux limites de leurs utilisations.

L'OTB nous apporte un panel d'algorithmes : les traitements *Mean-Shift* nous offrent des résultats convenables avec une bonne détection des pierres, l'algorithme *Watershed* est prometteur si nous le modifions un peu (ex : suppression des régions parasites par l'ajout d'un paramètre de taille minimale) et le dernier, *Morphological profiles based segmentation*, permet une très bonne détection des contours des pierres.

Cependant, cette bibliothèque n'a pas que des avantages. Nous avons rapidement remarqué durant les tests que les algorithmes demandent de bonnes ressources informatiques pour fonctionner (en particulier l'algorithme *Morphological profiles based segmentation)*.

Autre limitation flagrante : les images à traiter sont limitées en taille de pixel (Taille max. : environ 1200*1200 px). Les sources (questions posées sur un forum dédié à OTB) nous indiquent que cette restriction est liée à la mémoire de l'ordinateur disponible lors des traitements. Cependant, les tests ont été réalisés sur des ordinateurs ayant respectivement 4 et 8 Go de RAM et la restriction n'a pas évoluée. Il faudrait recommencer en recompilant les codes sur l'ordinateur utilisé.

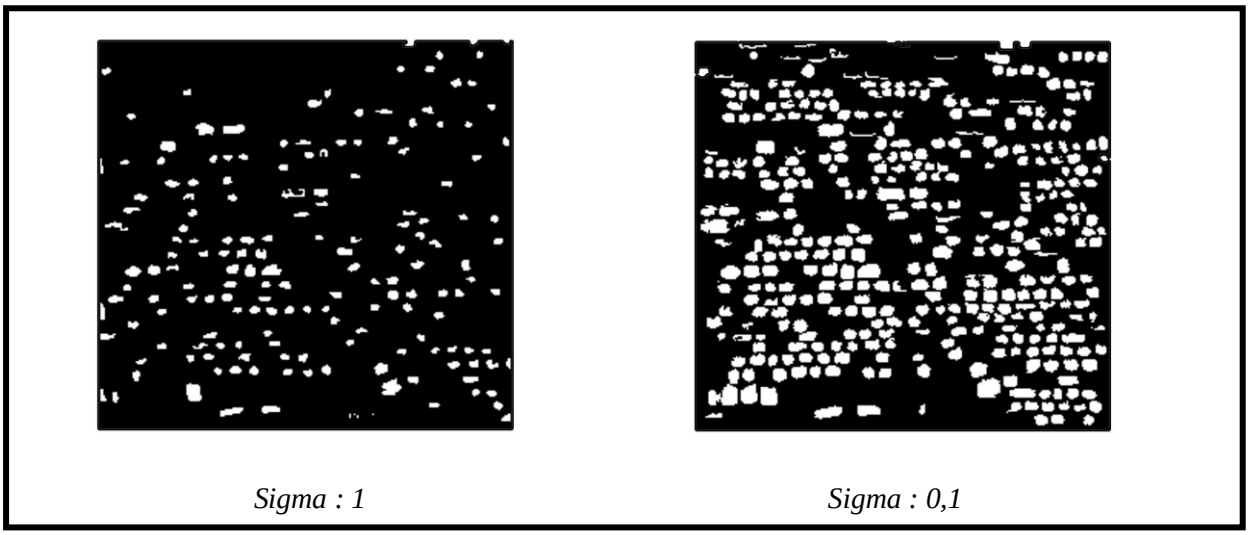
De plus, nous avons pu voir que les pierres de couleurs claires (blanches, beiges...) ne sont pas détectées. Le traitement repère parfois l'ombre provoquée par la pierre sur l'enduit.

Pour conclure, le problème de restriction de la taille des images à traiter est rédhibitoire. Nous ne pouvons pas nous permettre de continuer à travailler dans ce contexte alors que le commanditaire utilise des images plus lourdes (minimum 4-5000 x 4-5000 pixels).

3.3.4. Scikit

Après avoir analysé la bibliothèque de l'OTB, étudions Scikit et intéressons-nous à l'algorithme *Edge Based Segmentation*.

Cette méthode est ici fondée sur l'algorithme de Canny. Notre image étant peu bruitée, nous baissons la valeur du *sigma* (qui est par défaut de 1).



Sigma : 1 *Sigma : 0,1*

Figure 22 : Impact du paramètre *sigma* de l'algorithme *Edge based segmentation*

Nous obtenons un bien meilleur résultat (figure 22) en diminuant ce paramètre, les pierres détectées sont plus nombreuses et bien mieux délimitées.

Au final, le résultat est très propre avec une bonne délimitation des pierres détectées. Cependant, il subsiste des régions, liées aux ombres, à supprimer.

Explorons à présent l'univers du *Watershed* par l'algorithme de traitement nommé *Region based segmentation*. Nous pouvons jouer sur deux seuils pour classifier les pixels de l'image en fonction de leur valeur de niveau de gris. Le premier est lié à l'étiquetage de l'arrière-plan (seuil haut ici) et le second est lié aux régions d'intérêt (seuil bas ici). Plus on augmente le seuil des régions d'intérêt, et plus le résultat final aura de détails (figure 23).

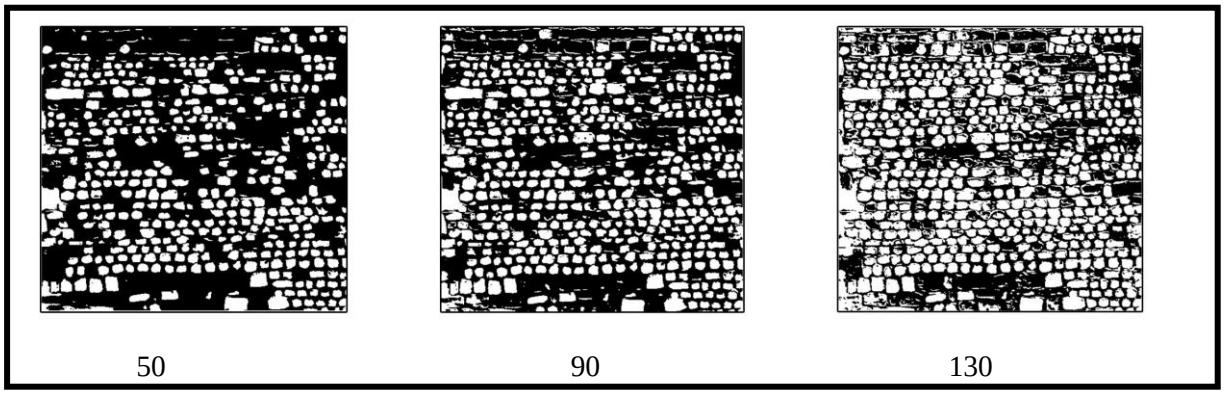


Figure 23 : Résultats selon le seuil des marqueurs des régions d'intérêt

Quantitativement, de nombreuses pierres seront détectées, cependant, qualitativement, beaucoup ne le sont pas bien. Regardons cela de plus près.

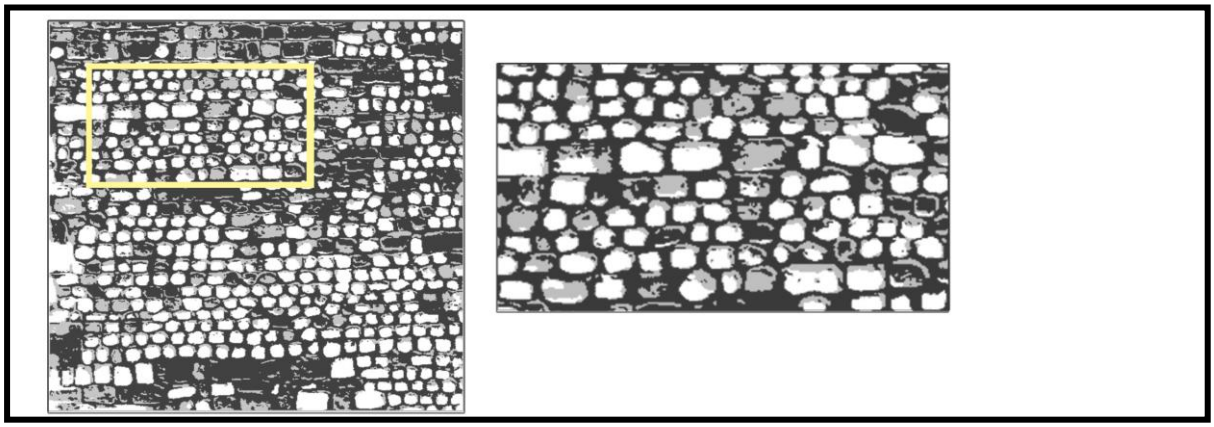


Figure 24 : Comparaison des résultats obtenus avec 50 (blanc) et 130 (gris) comme seuil bas

On observe une diminution (figure 24), en plus de celle du nombre de pierres détectées, du périmètre de ces pierres avec le seuil 50. Quel résultat est alors meilleur concernant l'emplacement des contours ?

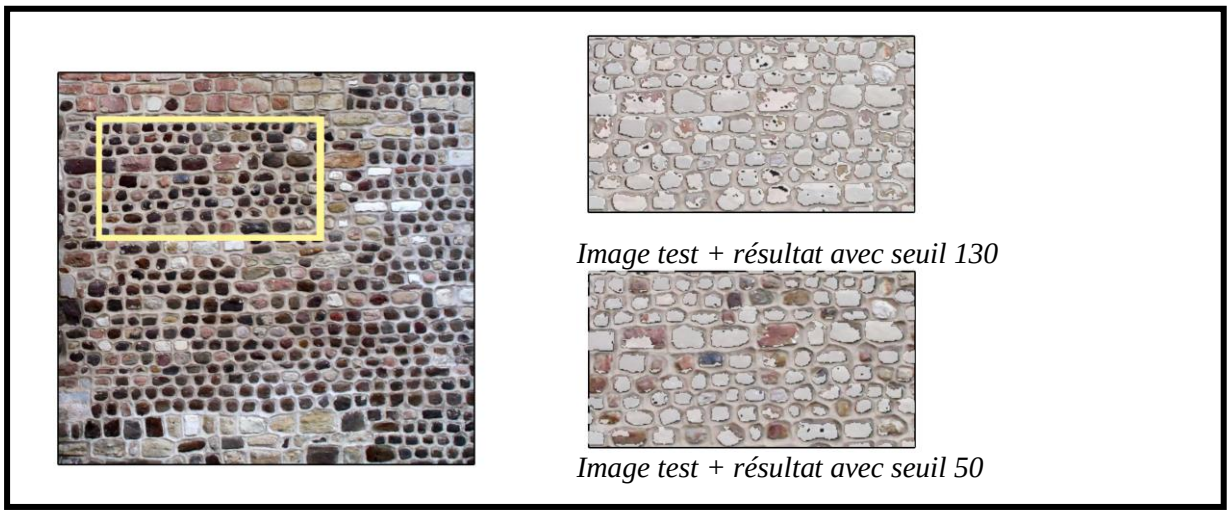


Figure 25 : Comparaison des résultats de la figure 20 (en gris), superposés sur l'image test

On remarque (figure 25) globalement que les contours des pierres sont mieux respectés lors d'un seuillage bas (ex : 50).

De plus, de nombreuses ombres sont détectées, ce que nous ne souhaitons pas, lors d'un seuillage haut (ex : 130).

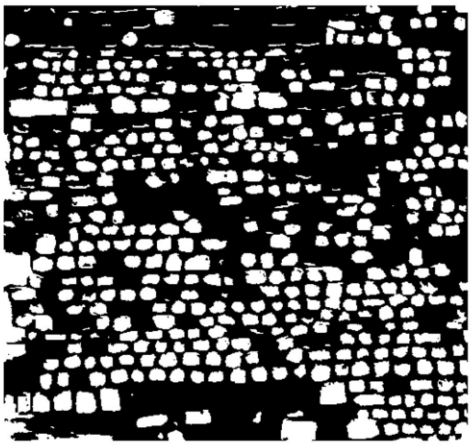
Quant au seuillage des marqueurs d'arrière plan, ce dernier travaille de la même façon que son homologue (plus le seuil sera restrictif, moins il y aura de pixels catégorisés comme « arrière plan », et donc plus il y aura de pierres détectées sur le résultat).

Attention :

- à ne pas descendre le seuil des marqueurs des régions d'intérêt trop bas sinon les contours seront trop restreints et deviendront faux (en plus du nombre de pierres détectées qui sera faible).
- à ne pas monter le seuil des marqueurs d'arrière plan trop haut sinon les pierres vont être confondues et le résultat sera erroné.
- à ne pas faire empiéter les seuils sur les mêmes valeurs de niveaux de gris (ex : 130 en seuil bas et 80 en seuil haut → la plage 80-130 est donc commune aux deux seuils), cela dégraderait le résultat.

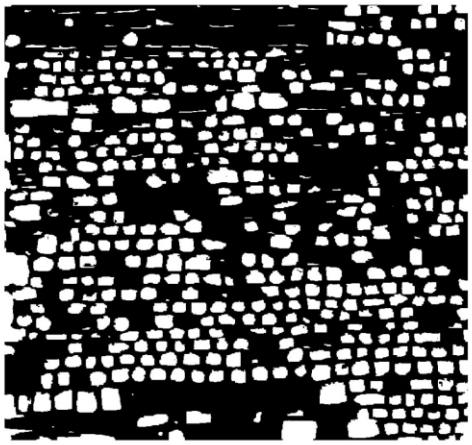
Intéressons-nous ensuite à l'algorithme *Random Walker*.

De façon similaire à la segmentation précédente, la méthode *Random walker* utilise des marqueurs pour détecter et dessiner les pierres. Nous obtenons des résultats très similaires entre les deux segmentations (figure 26).



Segmentation Region based

Seuil marqueurs « arrière plan » : 150,



Segmentation Random Walker

Seuil marqueurs « régions d'intérêt » : 50.

Figure 26 : Comparaison entre les deux algorithmes *Region based segmentation* et *Random walker* à travers leurs paramètres de seuil des marqueurs

Les résultats des tests sont les mêmes que pour la segmentation *Region based*.

Un des paramètres de la Segmentation *Random Walker* est *beta* : plus ce nombre est haut et plus la diffusion sera difficile avec des gradients élevés.

Ce paramètre semblait être intéressant, seulement, en l'état, il n'influe que très peu sur le résultat (figure 27).

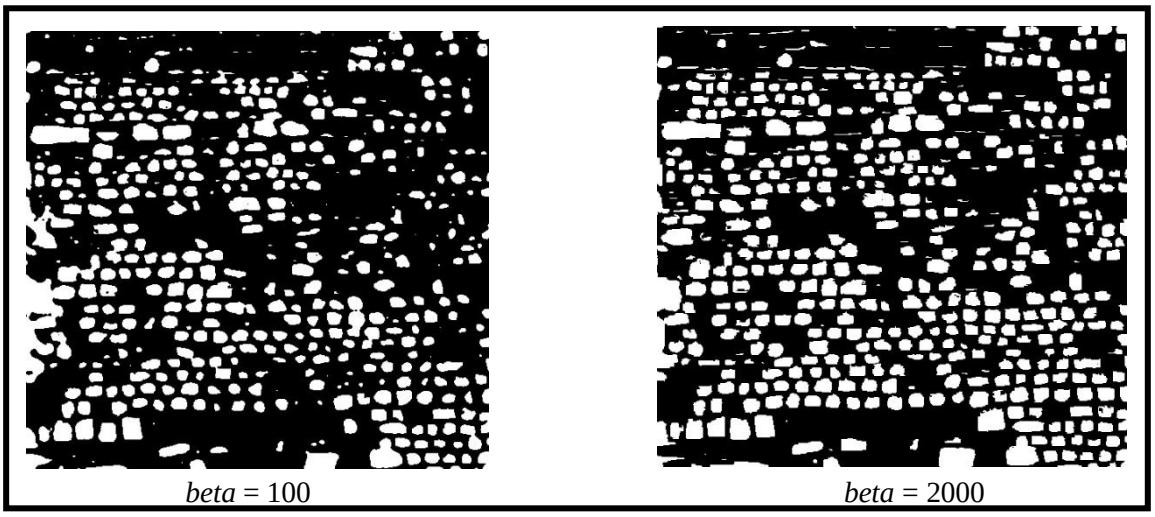


Figure 27 : Impact du paramètre *beta* de l'algorithme *random walker*

Parlons à présent de l'algorithme *Felzenszwalb*. Nous pouvons conclure directement que ce dernier n'est pas adapté à notre mode opératoire. Les résultats qu'il produit présentent un dégradé de niveaux de gris (figure 29) qui ne permet pas facilement de délimiter le fond et les pierres. Certes cela labellise l'image, ce qui est une bonne chose en soit, mais empêche la bonne lecture de l'image lors de sa conversion en vecteur dans ArcGIS (figure 28).

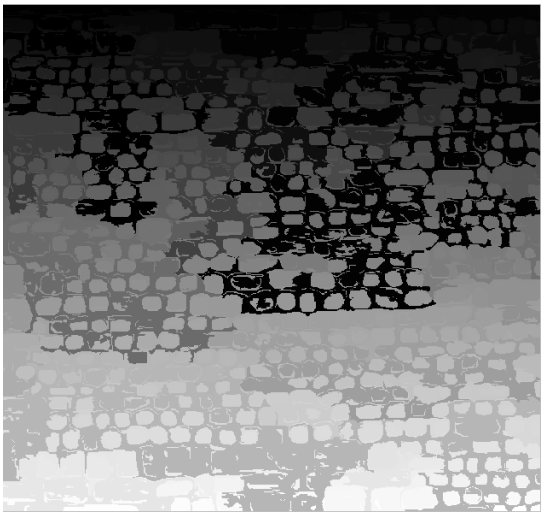


Figure 29 : Résultat de la segmentation *Felzenszwalb* (*scale* = 200, *sigma* = 1)

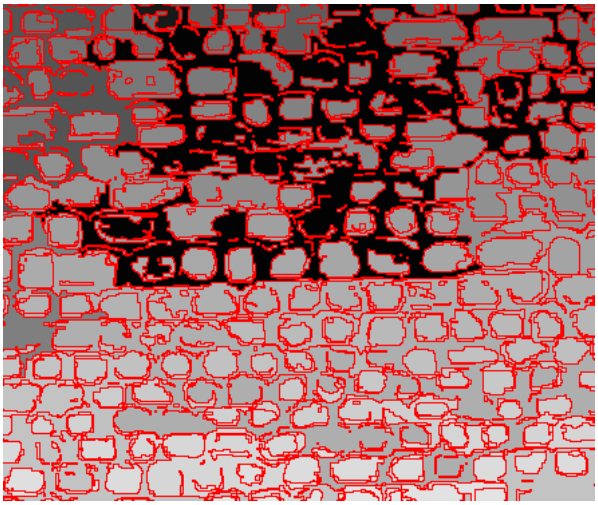


Figure 28 : Zoom sur le résultat superposé à sa conversion en vecteur sous ArcMap

Nous retenons de ces derniers tests les points suivants :

- L'algorithme *Edge Based Segmentation* trace des contours, lors de son traitement, puis ne garde que les contours fermés dans le résultat. Nous ne souhaitons pas cela car une majorité de pierres n'est pas détectée.
- Les algorithmes *Region based segmentation* et *Random Walker* utilisent tous les deux des seuils sur les niveaux de gris, ce qui n'est pas forcément bon car cela empêche de détecter par exemple les pierres claires si le seuil de régions d'intérêt ne sélectionne que les pixels de valeurs supérieures à 130. Malgré leurs limitations, ils offrent des résultats convenables avec cette image test.

- L'algorithme *Felzenszwalb* n'est pas adapté à notre mode opératoire et plus particulièrement lors de la conversion du raster en vecteur. Le résultat n'est donc pas utilisable.

Pour ces raisons, nous utiliserons l'algorithme *Region based segmentation* dans la suite.

3.3.5. Post-traitement

A la fin du traitement par notre algorithme *Region based segmentation*, nous obtenons un résultat binaire. Nous avons, entre autres, des petites régions détectées comme étant des pierres (alors que ce sont plus souvent des ombres) et des zones enclavées dans les pierres qui, par une différence de couleur trop importante avec le reste des pierres, ont été classées comme de l'arrière-plan (figure 30).

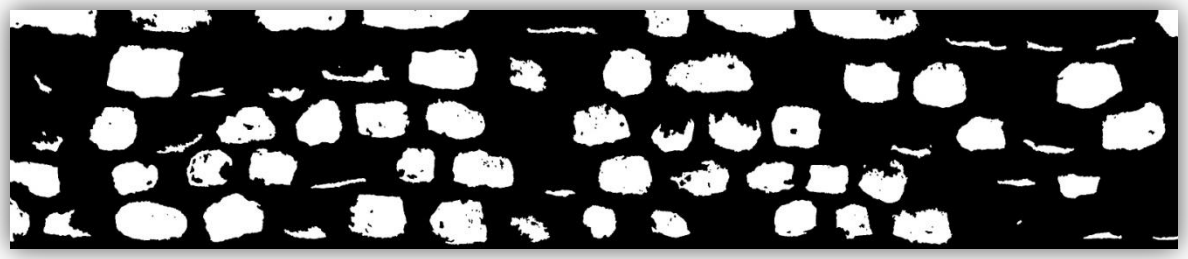
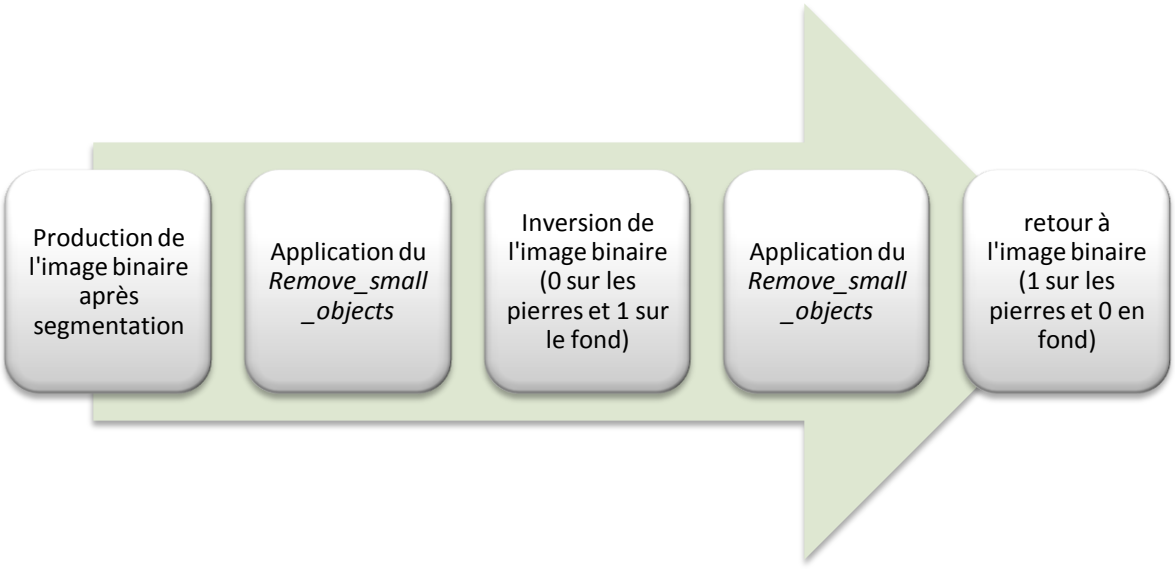


Figure 30 : Extrait d'un résultat après segmentation *Region based*

Nous cherchons à remédier à ce problème par un post-traitement : l'utilisation de la fonctionnalité *remove_small_objects*. Cette dernière, permet à partir d'une valeur fixée par l'utilisateur de rechercher dans l'image toutes les régions qui sont formées à partir d'un nombre de pixels inférieur à ce chiffre. Puis, elle supprime toutes ces régions. Comme nous l'avons dit, nous avons deux types de régions à supprimer. La seconde (les zones enclavées) possède des valeurs de pixels à 0. Or notre fonctionnalité ne recherche que les objets dont les pixels ont des valeurs supérieurs à 1. Nous avons donc procédé de la manière suivante :



Cela nous permet de retirer les petites zones « nuisibles » qu'elles soient de l'arrière-plan ou des régions d'intérêt (figure 31). Bien sûr, il en reste toujours mais cela dépend uniquement du seuil que l'utilisateur choisit.

Figure 31 : Extrait d'un résultat binaire après segmentation *Region based* et application du *Remove_small_objects*

3.3.6. Etude de l'Algorithme retenu

Lors de ces tests nous avons travaillé avec l'image test et l'image de la figure 1.

Nous avons ré-échantillonné ces deux images à différentes tailles de pixels pour appliquer à chaque résultante nos tests décrits en 3.3.2. . Par ce procédé, nous avons cherché à déterminer une taille de pixel relativement idéale (taille de pixel optimale pour un traitement automatique le meilleur qui soit).

Lors des tests, nous avons procédé au traitement d'image via l'algorithme sélectionné précédemment le *Region based segmentation*. Les résultats ont été produits sans l'option *remove_small_objects* puis avec, pour nous permettre de comparer une fois de plus l'impact qu'il peut avoir sur la qualité des résultats.

Concernant la première image test, nous avons fixé les paramètres de seuils d'arrière-plan à 50 et de région d'intérêt à 130 (figure 32).

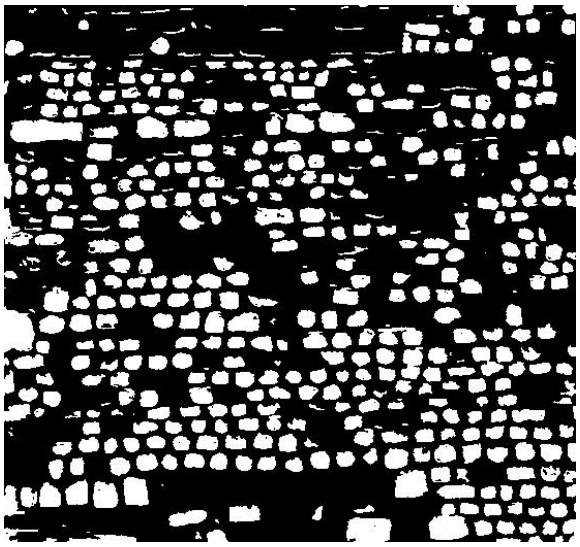


Figure 32 : Résultat obtenu après application, sur la première image test, de l'algorithme *Region based segmentation* avec respectivement 50 et 130 comme valeurs des seuils de marqueurs d'arrière plan et de région d'intérêt

Nous obtenons les résultats suivants concernant cette image et celles sous-échantillonnées qui en découlent :

Taille pixel (mm)	Taille approx. Objet (px)	Résultat indice 1	Résultat indice 2
0,7	15029	54,04	1,67
1,4	3757	51,90	1,92
2,8	939	45,34	3,62
5,6	235	41,88	4,54

On constate que mieux le raster est défini (taille de pixel faible), mieux les pierres seront détectées. Les 2 indices confirment cette hypothèse dans le cas de cette image.

Concernant la seconde image, nous avons fixé les paramètres de seuils d'arrière-plan à 90 et de région d'intérêt à 140. On obtient un résultat satisfaisant du traitement automatique vue l'image binaire produite après segmentation (figure 33).

Figure 33 : Résultat obtenu après application, sur la deuxième image test, de l'algorithme *Region based segmentation* avec respectivement 90 et 140 comme valeurs des seuils de marqueurs d'arrière plan et de région d'intérêt

Nous obtenons les résultats suivant concernant cette image (ainsi que ses homologues sous-échantillonnés) :

Taille pixel (mm)	Taille approx. Objet (px)	Résultat indice 1	Résultat indice 2
0,6	124476	84,60	29,90
1,2	31119	88,63	28,95
2,4	7780	88,64	28,01
4,8	1945	91,62	27,52
9,6	486	90,71	28,23

Les données ne décèlent qu'une pierre non détectée. Cela est dû au fait que les pavés sont assez bien définis (contraste important entre un pavé et le sol). De plus, les indices numéro 1 et 2, nous indiquent que les résultats s'améliorent lorsque l'on sous-échantillonne l'image jusqu'à 0,125 mais qu'ils se dégradent au delà. Cela nous laisserait penser que la taille de pixel optimale pour un traitement qui soit le meilleur possible (concernant cette image test) se trouverait aux alentours de 4,8 mm.

Néanmoins, si nous comparons les résultats des deux images, nous nous apercevons qu'il n'y a pas de taille optimale universelle de pixels. Nous ne l'avons pas trouvé pour la première image test mais cette dernière, dont les contours des pierres sont très bien définis, peut être très bien résolue (taille de pixel inférieure à 1 mm). Quant à la seconde image, dont les pollutions sont nombreuses (graviers et terres dont les variations provoquent des éclaircissements et des ombres, ce qui implique des gradients forts ; niveaux de gris divers au sein d'une même pierre...), le traitement n'est plus le même : il faut absolument baisser la définition de l'image pour limiter un maximum les gênes occasionnées par le sol (autre que pavé) sans pour autant que l'image soit si floue qu'on ne distingue plus correctement la limite des pierres.

Rappelons que ces résultats ont été produits sans la fonction *remove_small_objects*.

Par la suite, nous gardons les mêmes paramètres de segmentation pour chacune des deux images. Les résultats de la segmentation avec la fonctionnalité remove_small_objects sont les suivants pour l'image du mur de la cathédrale du MANS puis de l'image des pavés :

Taille pixel (mm)	Taille approx. Objet (px)	Résultat indice 1	Résultat indice 2	Résultat indice 3
0,7	15029	54,36	1,66	36,66 / 18,22
1,4	3757	52,53	1,83	35,14 / 24,52
2,8	939	46,32	3,74	13,23 / 32,75
5,6	235	44,77	4,12	9,98 / 37,31

Taille pixel (mm)	Taille approx. Objet (px)	Résultat indice 1	Résultat indice 2	Résultat indice 3
0,6	124476	88,65	31,50	50 / 2,94
1,2	31119	90,42	25,67	64,71 / 2,94
2,4	7780	90,81	25,44	70,59 / 2,94
4,8	1945	92,21	25,88	67,65 / 2,94
9,6	486	90,93	28,01	61,76 / 2,94

Ceux concernant l'image des pavés sont résumés dans le tableau suivant :

Les résultats obtenus pour chacune des images concordent avec ceux obtenus sans utilisation de la fonction *remove_small_objects* : les résultats sont les meilleurs pour l'image des pavés aux alentours d'un sous-échantillonnage de 0,125, pour l'image du mur, ils le sont pour une taille de pixel au moins inférieure à 0,7 mm.

On ressent particulièrement l'effet de l'ajout de la fonction dans ces derniers résultats : certains chiffres sont fortement biaisés.

Dernier point notable, le passage de la résolution de 1,4 mm à 2,8 mm implique une forte dégradation des résultats concernant les tests de l'image du mur de la cathédrale du MANS.

4. Développement d'une interface graphique

Concernant cette première partie du projet, (détection des contours de pierres et briques sur un mur de façade par la segmentation d'une image numérique) nous n'avons pas trouvé d'outil dédié à une telle démarche.

Nous avons fait le choix suivant : créer par nos propres moyens, à l'aide du langage de programmation Python, un outil possédant une interface graphique avec laquelle l'utilisateur peut interagir.

L'écriture de ce programme comporte 2 parties :

- la programmation de l'interface graphique,
- la compréhension et l'implémentation informatique de l'algorithme de traitement d'image.

4.1. Interface graphique

Concernant la réalisation de l'interface graphique, nous avons choisi le langage de programmation Python. Il existe au sein de ce dernier diverses bibliothèques de widgets[5] pour les interfaces. La plus courante et la plus simple d'utilisation est Tkinter. Le choix possible, plus complet et plus complexe, de la bibliothèque PyQT (simple lien entre Python et QT[6]) n'a pas été retenu puisque nous n'avons pas besoin des fonctionnalités supplémentaires que peut proposer PyQT.

De plus, durant un certain temps, nous pensions pouvoir nous contenter de l'interface déjà programmée via la bibliothèque de MatPlotLib (figure 34).

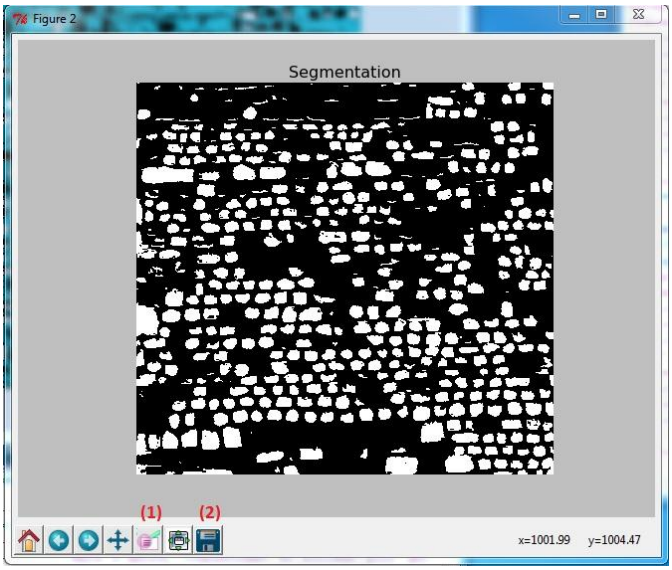


Figure 34 : Capture d'écran de l'interface graphique générée par la bibliothèque MatPlotLib

[5] Elément visuel d'une interface graphique déjà programmé dans une bibliothèque. Terme provenant des mots window (fenêtre) et gadget.

[6] QT, développée en C++ par QT Developpment Frameworks, offre différents composants d'interface graphique, d'accès aux données, de connexions…

Cette interface paraissait parfaite puisqu'elle nous permettait de visualiser et d'enregistrer nos résultats (2) très facilement. De plus, elle bénéficiait d'un zoom (1). Cependant, nous nous sommes rapidement rendu compte que l'interface enregistrait le résultat qu'elle affichait (ce dernier étant dégradé pour être affiché à l'écran). Nous avons alors écarté cette solution, même si lors de nos tests des algorithmes, elle nous a été d'une grande aide car rapide d'utilisation (peu de programmation pour visualiser nos résultats).

Commençons par découvrir la conception de la fenêtre principale qui a été développée avec Tkinter (figure 35).

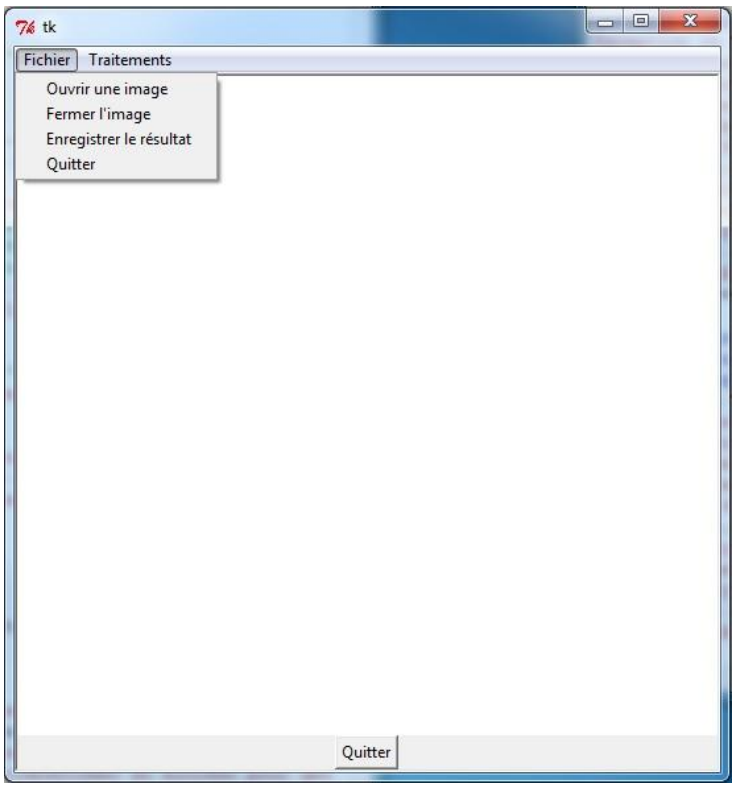


Figure 35 : Capture d'écran de la fenêtre principale du programme développé en Python

Elle dispose tout d'abord de deux menus en haut à gauche.

Le premier, nommé « fichier », comporte 4 sous-menus.

Le premier nous permet simplement d'ouvrir l'image raster que nous souhaitons traiter. Pour cela il utilise l'explorateur Windows habituel. De nombreux types d'images sont assimilables. Néanmoins, nous préconisons d'utiliser les fichiers d'image TIFF[7] si c'est une orthophoto (le programme reconnait les fichiers de calage .TFW[8] et émettra automatiquement celui du résultat lorsqu'il sera enregistré).

Une fois l'image ouverte, la fenêtre graphique se modifie et affiche cette image. L'image affichée n'est pas celle qui sera traitée puisque celle qui apparaît est rééchantillonnée pour ne pas dépasser 500px de hauteur tout en gardant ses proportions. Ainsi on évite d'obtenir une fenêtre trop encombrante à l'écran.

[7] Tagged Image File Format
[8] « Tiff World File » : format de fichier composé de 6 paramètres permettant de géoréférencer par 4 transformations (rotation, mise à l'échelle, translation et retournement) l'image concernée

Dans le cas où nous souhaitons ne plus interagir avec cette image, le deuxième sous-menu nous permet de fermer l'image ou d'en ouvrir une nouvelle (ce qui écrasera la précédente).

Le troisième sous-menu nous permet d'enregistrer le résultat après traitement et validation de ce dernier. Une première fenêtre (image du haut, figure 36) s'ouvre et nous permet de choisir l'emplacement où sera enregistré le raster résultat. Puis une seconde fenêtre (image du bas, figure 36) s'ouvre après que nous ayons validé l'emplacement du fichier d'enregistrement. Cette dernière nous propose d'écrire le nom du fichier que nous allons enregistrer avec un nom par défaut « résultat.tif ». Il suffit d'un clic pour terminer la session.

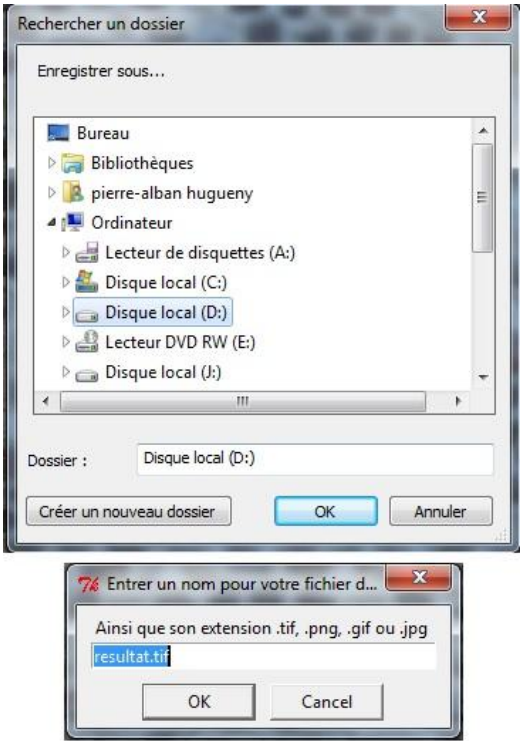


Figure 36 : Captures d'écran des boîtes de dialogue permettant l'enregistrement des résultats produits après traitement

Concernant le quatrième sous-menu, il permet de quitter l'outil et son interface. Lorsque l'on clique dessus, une fenêtre de confirmation (figure 37) nous demande si nous sommes certains de vouloir fermer, simple marge de sécurité dans le cas d'un clic de souris mal placé. Nous retrouvons le même système de fermeture en cliquant sur le bouton « Quitter » au centre-bas de la fenêtre principale.

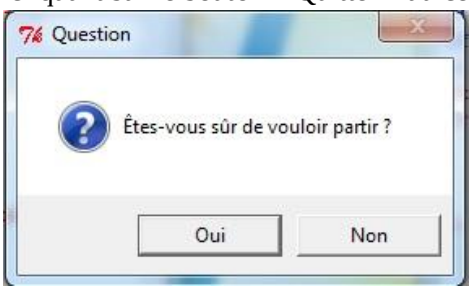


Figure 37 : Capture d'écran de la fenêtre secondaire permettant de ne pas quitter le programme sans sécurité

Quant au deuxième menu, nommé Traitements, il ne dispose que d'un sous-menu actif. Ce sous-menu nommé *Region Based Segmentation* nous mène à l'application de l'algorithme de segmentation de l'image raster ouverte en entrée. Pour cela, il nous demande, par une succession de boîtes de dialogue (figure 38), d'entrer les différentes valeurs des paramètres de l'algorithme.

Figure 38 : Capture d'écran d'une des boîtes de dialogue permettant de saisir en entrée du traitement les valeurs des paramètres de l'algorithme

Après avoir validé ces paramètres, le traitement est lancé. A la fin de ce dernier, le résultat produit apparaît à l'écran dans une fenêtre de MatPlotLib. Si le résultat convient, il suffit de retourner sur le sous-menu en question pour enregistrer le résultat. Sinon, nous pouvons relancer le calcul en changeant les paramètres.

En plus de la programmation de notre fenêtre principale, il a fallu programmer l'affichage des boîtes de dialogue. Elles restent pour le moment simples puisque ce sont des fonctions de la bibliothèque Tkinter. Mais pour une question d'ergonomie, il serait bon de les revoir : lors du choix des paramètres de segmentation, il s'enchaîne plusieurs boîtes de dialogue. Nous avons pensé à les regrouper en une unique boîte (figure 39), mais cela n'a pas pu être réalisé à cause de problématiques informatiques.

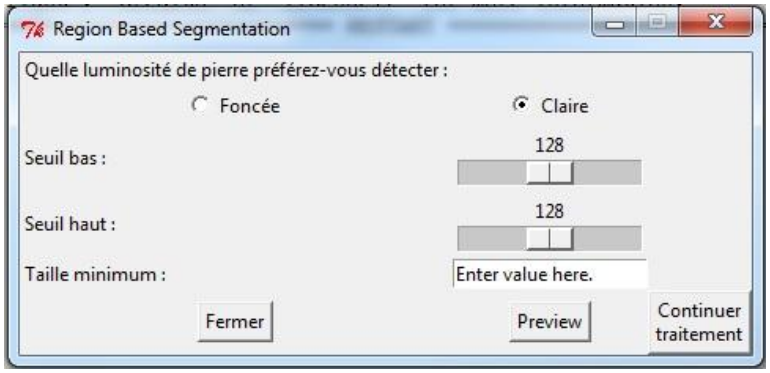


Figure 39 : Boîte de dialogue unique

Pour terminer, nous avons mis en place un système de gestion des erreurs au sein du programme :

A divers endroits, nous avons placé dans le code des gestions des exceptions par l'utilisation de « try : » et « except : » (figure 40) qui marchent de pair. Dans le cas où la partie du programme compris dans le « try : » rencontre des problèmes, alors un message généré par le « except : » est affiché à l'utilisateur pour le prévenir d'une erreur. Ainsi, l'utilisateur peut savoir que sur une partie précise du programme, il y a eu un problème. Mais attention cela n'est pas forcément dû à une erreur de programmation, cela peut être aussi dû à une mauvaise manipulation de l'utilisateur (message vert, figure 40).

```
#-------------------------------------------------
    def fermer(self):
        try:
            self.can.delete(ALL)
        except:
            messagebox.showwarning('Attention', "Pour fermer une image, il faut d'abord l'ouvrir.")

#-------------------------------------------------
```

Figure 40 : Code Python version 3.3 de la fonction « fermer » permettant de supprimer une image ouverte dans le programme

4.2. Implémentation du traitement d'image

Entre le moment où l'on ouvre une image, le moment où on applique le traitement d'image et le dernier moment où l'on enregistre notre résultat, notre image peut changer très souvent de type (codages flottant, entier, booléen...). La gestion de ces images est un point problématique que nous avons réglé à travers l'utilisation de différentes bibliothèques : la bibliothèque Scikit-Image permettait de traiter l'image et la bibliothèque *Pillow* permettait l'ouverture, l'enregistrement des images ainsi que la modification de leur type.

De plus, on peut ouvrir une image avec une certaine fonction d'une bibliothèque donnée puis une fonction suivante, d'une autre bibliothèque, nécessite en entrée un type précis d'objet qui n'est pas celui fourni par la première fonction. Par exemple, les fonctions de la bibliothèque Scikit ne peuvent ouvrir que des images au format JPEG alors que nous souhaitons pouvoir accéder à des données du type .PNG ou encore .TIFF. Il a donc fallu utiliser la fonction d'une autre bibliothèque (Pillow = « PIL »), qui peut ouvrir plus d'une trentaine de type d'images différentes. Puis nous avons converti cette image dans un format que peuvent comprendre les fonctions de Scikit.

Autre exemple, la fonction *remove_small_object* qui permet de retirer de notre résultat les composantes faisant moins d'une certaine surface (en pixel) ne fonctionne qu'avec des données de type booléen.

Nous obtenons à la fin de notre traitement un tableau dont les cases (→ pixels si transformé en image) sont à 1 (ou 255) lorsque ce sont des pierres et 0 sinon.

Le résultat final du programme de segmentation automatique est un raster (image binaire).

5. Description vectorielle

Par la fonctionnalité d'Arcgis, nous convertissons l'image raster en un fichier vectoriel ne comprenant que les contours des pierres et briques. L'étape suivante consiste à modifier le fichier vectoriel pour qu'il devienne l'équivalent d'une « vérité terrain » (c'est-à-dire corriger les contours que l'utilisateur considère faux). S'offrent alors à nous plusieurs choix : nous utilisons soit Arcgis, soit Inkscape soit nous produisons notre outil par nos propres moyens en Python. Nous avons écarté la dernière solution pour cause de temps, produire un tel outil aurait été chronophage alors qu'il existe déjà des programmes le permettant : autant se concentrer sur des points plus importants. Le CAPRA utilisant déjà le pack Arcgis pour dessiner ses vérités terrains à la main, on opta pour cette solution.

5.1. Transformation raster ↔ vecteur

Le logiciel ArcMap bénéficie de ce qu'on appelle des « ArcToolBox » (ou Boîte à outils). Ces dernières représentent la majorité des fonctions de traitement présentes dans le logiciel. Entre autres, dans la catégorie « Outils de conversion », deux sous-parties nous intéressent particulièrement : « A partir d'un raster » et « vers raster ». On y trouve respectivement, l'outil « raster vers polygones » permettant de convertir un fichier raster en un fichier vectoriel puis l'outil « polygone vers raster » permettant de faire l'opération inverse.

Concernant le premier outil, il faut tout d'abord choisir le raster à convertir. Après avoir choisi notre fichier, il apparaît les champs possibles (souvent « Value » et « Count »). Comme précisé, le choix ici est facultatif puisque, que l'on choisisse l'un ou l'autre, la table attributaire du fichier résultat comprendra les deux champs. Pour terminer, nous avons le choix de simplifier ou non les polygones. Si ces derniers ne sont pas simplifiés, la limite suivra scrupuleusement la délimitation des pixels (figure 42), sinon elle sera simplifiée (figure 41).

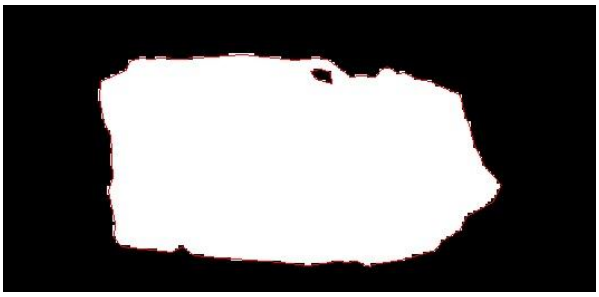


Figure 42 : Zoom sur la vectorisation d'une pierre après conversion du raster (option simplification des polygones)

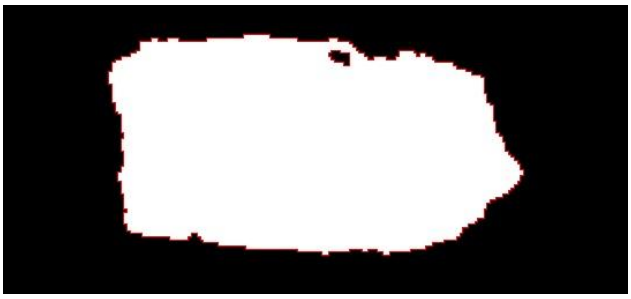


Figure 41 : Zoom sur la vectorisation d'une pierre après conversion du raster sans option

Lorsque l'on clique sur « OK », le résultat est alors produit, enregistré et affiché dans ArcMap.

Quant au second outil, il permet beaucoup plus d'options. Comme précédemment, nous entrons en premier le fichier vecteur puis nous choisissons le champ de données (le plus souvent utilisez ici le champ *Gridcode*). Mais cette fois-ci, le choix du champ est très important. Lors du dessin du raster, ce sont les valeurs du champ sélectionné qui vont être attribuées à chacun des pixels. Pour terminer, nous devons fixer 3 paramètres. Le premier est la méthode d'attribution des valeurs à chaque pixel. Dans le cas ou nous n'aurions pas coché la simplification des polygones lors de la création du fichier vectoriel, cette option est facultative. Sinon, voici la différence explicite entre les 3 choix possibles:

- « *CELL_CENTER : Le polygone qui se superpose au centre de la cellule détermine l'attribut à affecter à la cellule.*
- *MAXIMUM_AREA : L'entité unique qui présente la plus grande surface à l'intérieur de la cellule détermine l'attribut à affecter à la cellule.*
- *MAXIMUM_COMBINED_AREA : Si la cellule contient plusieurs entités de même valeur, les surfaces de ces entités sont combinées. L'entité combinée à la plus grande surface contenue dans la cellule détermine la valeur à attribuer à la cellule.* » (ESRI, 2013)

Le second paramètre est le champ de priorité : dans le cas où deux entités se superposent, il détermine celle qui aura la préséance.

Le dernier paramètre est la taille des cellules, il suffit d'indiquer la taille de nos pixels souhaitée en mètre.

Ces outils permettent de passer rapidement d'un vecteur à un raster et vice versa mais ne provoquent-ils pas une diminution de la précision générale du fichier ?

Pour vérifier cela, nous avons effectué l'expérience suivante : à partir d'un raster produit par notre programme Python (figure 42), nous avons converti celui-ci en un fichier vectoriel par le premier outil puis nous avons converti ce dernier résultat en raster par le second outil (figure 43). Pour conclure, nous avons comparé le raster final à celui de départ. Notez que la case « simplifier les polygones » n'a pas été utilisée.

On peut voir qu'il n'existe aucune différence entre les figures 43 et 44.

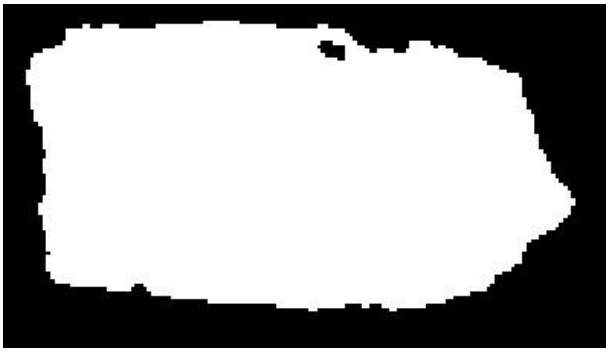


Figure 44 : Zoom sur le raster initial, sans post-traitement après la détection des contours

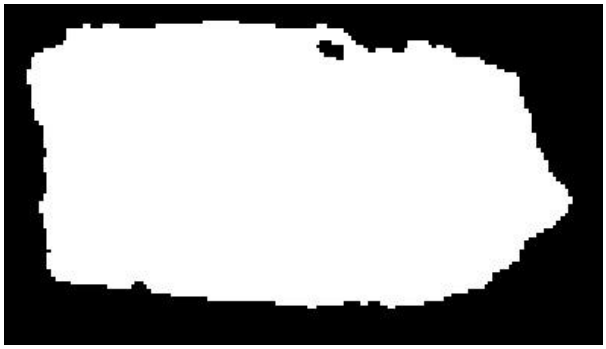


Figure 43 : Zoom sur le raster obtenu à partir d'une conversion d'un vecteur, lui-même obtenu à partir du raster initial

Il n'y a donc aucune perte de précision si nous procédons de cette manière.

Recommençons ce test mais cette fois-ci avec l'option « simplifier les polygones » du premier outil puis en laissant « CELL_CENTER » pour le choix d'attribution des valeurs aux cellules dans le second outil.

La transformation ne change rien, les pixels ont toujours les mêmes valeurs au départ comme à la sortie (figure 46).

Testons à présent l'option « MAXIMUM_AREA » comme paramètre (figure 45). Les résultats bien que très similaires ne sont plus les mêmes.

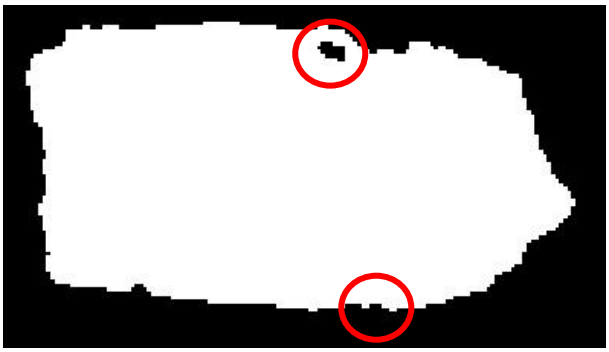


Figure 46 : Zoom sur un raster converti à partir d'un vecteur en utilisant l'option *Cell_center*

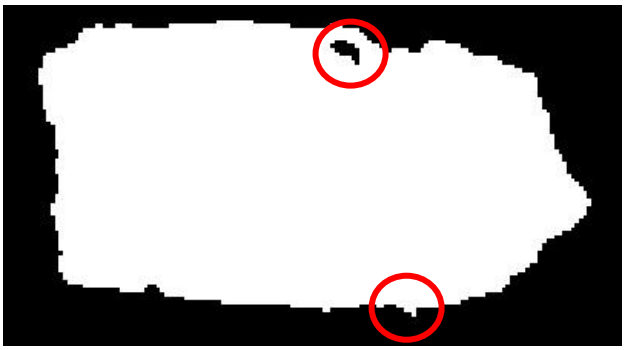


Figure 45 : Zoom sur un raster converti à partir d'un vecteur en utilisant l'option *Maximum_area*

5.2. Modification du vectoriel

L'utilisateur cherche, après génération automatique d'un résultat et conversion en vectoriel, à corriger les erreurs commises par l'interprétation de l'algorithme de traitement d'image. Pour cela, il va devoir modifier les contours vectoriels (supprimer, rajouter des points, couper...).

Par Arcgis, la modification d'un fichier vectoriel passe par la barre d'outils « Editeur » (figure 47).

Figure 47 : Capture d'écran de la barre d'outil « Editeur » du logiciel Arcmap

Cette dernière se décompose en 4 menus :

- le menu d'édition nous permettant de commencer une session de mise à jour (1),
- les outils de sélection (2),
- les outils de dessin (3),
- les outils de modification (4).

Dans Arcgis, pour modifier une table, que ce soit un simple attribut ou une géométrie (coordonnées des points de contour), il faut démarrer une session de mise à jour. C'est là qu'intervient notre premier outil : le menu d'édition.

Ensuite, nous avons les outils de sélection nous permettant de sélectionner quel contour de pierre est à modifier.

Quant aux outils de dessin, ils nous permettent de tracer diverses formes (arcs, cercles, angle droit …) mais compte tenu de la précision attendu lors du dessin des contours et de la diversité de forme parmi les pierres, le simple outil « Segment droit » nous convient. Cet outil fonctionne de la manière suivante : à chaque « clic » de souris, un sommet est représenté puis entre chaque sommet un segment droit est dessiné. Ainsi, plus il y aura de sommets et plus les contours seront finement reproduits.

Concernant les outils de retouche, trois nous intéressent plus particulièrement :

- L'outil de modification des sommets qui nous permettra de changer l'emplacement d'un sommet,
- L'outil de remodelage qui, après sélection d'une entité, nous permet de tracer une polyligne (spécifier par des points) et redécoupera notre forme en fonction de cette polyligne : la partie de plus petite taille sera supprimée. Cet outil nous intéresse plus particulièrement lorsque le contour d'une pierre englobe des zones non désirées comme l'ombre de la pierre sur le joint,
- Le dernier outil est celui de découpe. En traçant une polyligne à travers une forme, l'entité sera simplement divisée. Cela créera donc une nouvelle ligne dans la table attributaire pour la nouvelle forme (et mettra à jour celle de l'ancienne). Cet outil nous intéresse particulièrement lorsque plusieurs pierres sont détectées par l'intermédiaire d'un seul et unique contour.

6. Classification des pierres en 7 catégories

Avec l'aide du CAPRA, nous avons pu établir une liste de 7 catégories de pierre. Ces 7 catégories représentent la majorité des pierres que peut rencontrer l'association du CAPRA lors de l'étude des maçonneries locales.

6.1. Présentation des catégories de pierre et extraction de leurs caractéristiques

L'homme, dans son processus de reconnaissance des pierres, fait l'analogie entre certains aspects de la pierre considérée et des caractéristiques des catégories de pierre. Procéder ainsi lui permet de savoir à quelle catégorie la pierre correspond.

Dans notre classification automatique des pierres, nous avons cherché à simuler ce que l'œil de l'homme pouvait interpréter. Ce dernier, en voyant une pierre, observe la pierre sur différents points : sa forme, son aspect ou encore sa couleur. Nous avons donc souhaité étudier les pierres sur ces différents points et en avons tiré, pour terminer, des critères spécifiques à chacune des catégories de pierre. ArcMap offre un outil permettant facilement de produire des statistiques sur la forme des pierres (aire, périmètre, valeurs de la meilleure ellipse englobante…). Pour cela, il nécessite de connaître la taille des pixels afin de procéder aux calculs. Mais si nous souhaitons des résultats corrects, il nous faut connaître la taille de pixel de nos échantillons, ce qui n'est pas le cas !

Pour commencer, nous avons, grâce au CAPRA, obtenu des imagettes[9] représentatives de chaque catégorie (figure 4). Nous avons commencé par étudier les canaux R, G, B des pierres puis nous avons continué sur les canaux H, S, V. Pour chaque canal, nous avons étudié le comportement des valeurs minimales, maximales, de la moyenne des valeurs des pixels dans chaque pierre, ainsi que des écarts-types. Rappelons que les canaux R, G, B et H, S, V sont directement liés. Cependant, les canaux H, S, V peuvent faire apparaître certains aspects des pierres que ne pourraient pas révéler les canaux R, G, B et vice versa.

Nous pouvons trouver en annexe n°3 à n°16 les tableaux résultants de ce travail. Les tableaux sont indexés dans l'ordre alphabétique des pierres, et pour chacune d'elle, on trouve les mesures prises concernant les 6 canaux cités précédemment.

6.2. Critères d'appartenance à une catégorie et règles de classification

Pour établir les critères de chaque catégorie de pierre, nous avons observé (sur les échantillons) l'étendue générale des valeurs des paramètres concernés.

Prenons l'exemple du canal rouge de la brique (annexe n°3, canal rouge). On observe que les valeurs moyennes sont étalées entre 98 et 133. Ces extrêmes forment la plage du critère « Rouge » de la catégorie « Brique ».

Nous établissons ces plages pour chacune des 7 catégories de pierre. Ensuite, nous procédons en suivant une règle des probabilités : la somme des probabilités d'appartenance pour toutes valeurs

[9] Ces imagettes sont des images relativement petites, comprenant uniquement la pierre à considérer et de préférence en plusieurs exemplaires pour pouvoir établir des statistiques

d'un critère doit être égale à 100 %. Ainsi, si la valeur moyenne d'un critère d'une pierre n'appartient qu'à une plage parmi les catégories de pierres, il y aura, 100 % de chances que la pierre appartienne à la catégorie en question. De plus, si une valeur occupe deux plages soit deux catégories, il y aura 50 % de chances que la pierre appartienne à la première ou à la deuxième catégorie. Et ainsi de suite …

De plus, les 7 catégories de pierre n'occupant pas l'ensemble des valeurs, nous avons établi une huitième catégorie de pierre que nous appelons « Autres ». Cette dernière n'a en soit aucune valeur, critère d'appartenance fixé. Elle ne fait que remplir les « espaces vides » pour respecter notre loi liée aux statistiques (somme des probabilités toujours égale à 100%). Ainsi si une pierre possède une valeur qui ne se trouve être dans aucune des plages des 7 catégories, alors cette pierre (pour le critère en question) aura 100% de chances d'appartenir à la catégorie « Autres ».

Dès qu'une valeur d'une pierre se trouve être en dehors de la plage d'une catégorie, sa probabilité d'appartenance à cette catégorie passe directement à 0. Il serait intéressant d'étudier une autre solution pour qu'elle ne réagisse pas aussi radicalement (en prenant par exemple une pente avec une certaine valeur pour passer à 0). Par manque de temps, nous n'avons pas pu aller aussi loin et avons fait le choix de rester simple pour voir l'ensemble du projet.

6.3. Implémentation des critères dans Arcgis à travers la classification automatique

Nous avons ici développé 2 outils (reliés à des scripts python version 2.7) permettant la classification automatique.

Le premier devra :

- Recevoir en entrée, par l'interface de la boîte d'outil d'ArcMap, le raster de l'image R,G,B ou H,S,V ainsi que le vecteur corrigé par l'utilisateur,
- Séparer les rasters pour obtenir les 3 canaux individuellement,
- Produire les statistiques zonales sur chacun des 3 canaux. 3 tables sont alors produites et ouvertes dans ArcMap,

Le second outil devra :

- Pouvoir naviguer à travers les 3 tableaux produits pour en tirer des informations,
- Comparer ses valeurs aux plages déterminées dans le chapitre 6.1. et obtiendra des probabilités d'appartenance à chaque catégorie de pierre,
- Après que ses comparaisons soient terminées, les additionner puis diviser le tout par le nombre de paramètre total (soit actuellement 3) pour obtenir la probabilité totale d'appartenance à chaque catégorie de pierre,
- Créer une colonne « Rslt Class. » (= résultat classification) dans la table du vecteur,
- En fonction de la plus grande des probabilités d'appartenance, attribuer à chaque pierre sa catégorie en écrivant le numéro de la catégorie (ex : 1 ou 2) dans la colonne « Rslt Class. » créée précédemment.

Ces numéros sont attribués par ordre alphabétique à chaque catégorie, soit :

- 1 = la brique,
- 2 = le calcaire de bernay,
- 3 = le granite,
- 4 = le grès cénomanien,

- 5 = le grès éocène,
- 6 = le grès roussard,
- 7 = le tuffeau.

6.4. Résultats

Pour terminer cette partie par des tests, nous allons produire des classifications en fonction de R, G, B seulement puis seulement à partir de H, S, V pour voir quelle est la meilleure solution. Ces canaux étant liés mathématiquement, des informations sont redondantes. Pour comparer ces résultats, nous nous appuyons sur la vérité terrain transmise par le CAPRA (classification de toutes les pierres de l'image du mur de la cathédrale du MANS dans une table).

Nous obtenons les résultats suivant (461 pierres au total) :

- 96 résultats correspondants (20,82 %) à la vérité terrain pour la classification à partir des paramètres H, S, V,
- 66 résultats correspondants (14,32 %) avec la vérité terrain pour la classification à partir des paramètres R, G, B.

Ces résultats ne sont pas encore très intéressants. Les lignes bleutées (figure 48) mettent en avant la bonne classification des paramètres R, G, B. De manière générale, nous remarquons que pour la classification à partir des canaux R, G, B, les pierres les mieux détectées sont les numéros « 2 » et « 6 » (soit les calcaires de bernay et les grès roussard). Quant à la classification à partir des canaux H, S, V, ce sont principalement les grès roussard qui sont détectés. Il faut surtout retenir la détection des calcaires de bernay par la première classification puisque les grès roussard représentent 77% des pierres du mur (selon la classification du CAPRA). Calculer des indices de forme que sont l'élongation, le facteur circulaire et le facteur rectangulaire à partir du tableau de statistiques géométriques, puis les comparer aux plages obtenus pour les catégories de pierre afin d'obtenir les probabilités d'appartenance, permettrait une meilleure classification mais nous n'avons pas eu le temps de mettre en place ces indices de forme.

FID	Shape *	OBJECTID	Id	gridcode	Shape_Leng	Shape_Area	Valeur	Valeur2	classe	classif	classhsv
197	Polygone	2346	204	0	0.38886	0.008508	255	130	2	2	5
198	Polygone	2347	205	0	0.615532	0.018277	255	130	2	2	4
199	Polygone	2348	206	0	0.279086	0.005057	255	130	6	1	1
200	Polygone	2349	207	0	0.56095	0.016323	255	130	2	4	5
201	Polygone	2350	208	0	0.263489	0.00496	255	130	2	2	2
202	Polygone	2351	209	0	0.358908	0.008639	255	130	6	1	1
203	Polygone	2352	210	0	0.383355	0.010053	255	130	6	3	3
204	Polygone	2353	211	0	0.344584	0.008923	255	130	6	1	8
205	Polygone	2354	212	0	0.352468	0.007836	255	130	6	6	3
206	Polygone	2355	213	0	0.26227	0.005176	255	130	6	1	1
207	Polygone	2356	214	0	0.311195	0.006796	255	130	6	1	3
208	Polygone	2357	215	0	0.327936	0.007763	255	130	6	1	6
209	Polygone	2358	216	0	0.282763	0.004068	255	130	6	1	6
210	Polygone	2359	217	0	0.282805	0.00578	255	130	6	5	1
211	Polygone	2361	219	0	0.311669	0.006724	255	130	6	5	1
212	Polygone	2362	220	0	0.302843	0.005288	255	130	6	1	6
213	Polygone	2363	221	0	0.307569	0.00539	255	130	6	1	1
214	Polygone	2364	222	0	0.263562	0.003988	255	130	6	6	5
215	Polygone	2367	225	0	0.365112	0.00911	255	130	6	1	6
216	Polygone	2368	226	0	0.377732	0.009345	255	130	6	5	1
217	Polygone	2369	227	0	0.319217	0.007301	255	130	2	4	5
218	Polygone	2370	228	0	0.339377	0.007622	255	130	6	1	6

Figure 48 : Extrait du résultat final de la classification. A partir de la droite : 1ère colonne classification HSV, 2ème colonne classification RGB, 3ème colonne classification vérité terrain

7. Conclusions

L'objectif de ce travail de fin d'étude était d'établir un outil permettant la détection automatique des contours de pierres d'un mur puis la classification automatique de ces pierres. Cet outil devait aussi permettre à l'utilisateur de retoucher les résultats produits puisque le traitement informatique ne pouvait être parfait. Cet outil, au final, devait permettre aux utilisateurs (archéologues) de gagner du temps puisqu'à ce jour, ils produisent ces résultats en les traitant entièrement manuellement sous informatique.

Premièrement, nous avons développé un outil dans le langage de programmation Python qui permet de détecter les contours des pierres. Nous avons implémenté comme méthode de segmentation le *region based segmentation* (~*Watershed*), un algorithme de traitement d'image sélectionné dans une des bibliothèques à notre disposition (libre d'accès). Ce dernier, après avoir acquis en entrée le fichier raster (image de la façade), génère un fichier raster de sortie dont les pixels sont caractérisés comme étant dans des pierres ou le contraire. Cet outil bénéficie, de plus, d'une interface graphique pour que l'utilisateur puisse dialoguer avec lui en toute simplicité.

Nous avons souhaité que l'utilisateur puisse retoucher ce résultat automatique. Nous avons donc pour cela utilisé le logiciel Arcgis et ses outils déjà configurés pour modifier des entités vectorielles. De plus, ce dernier offrait la possibilité de convertir des fichiers raster en vecteur.

Pour terminer, nous avons traité la classification des pierres dans notre projet. Il a été établi avec l'aide du CAPRA que ces derniers distinguaient 7 principales catégories de pierre dans leurs travaux locaux. Nous avons donc pu établir quelques caractéristiques propres à chacune d'elle pour en déduire des paramètres et des critères d'appartenance. Au final, nous obtenions une classification supervisée de chaque pierre via des règles d'appartenance.

Du point de vue du traitement de l'image, le sujet principal n'était pas de rechercher une solution optimale (algorithme(s) dédiés à la segmentation de pierres). Nous avons préférer étudier l'ensemble du projet, tout en n'approfondissant pas certain point tel la segmentation des pierres. Il reste donc de nombreuses recherches à produire de ce côté ci. Par exemple, notre algorithme sélectionné ne permet pas de détecter à la fois les pierres claires et les pierres foncées d'une image. Une solution à ce problème serait de mettre en place un système de quadrillage de l'image. Au début nous sélectionnons 3 fenêtres (l'une avec une pierre claire, l'une avec une pierre foncée et la troisième avec du fond comme seulement des joints), chacune de ces fenêtres dispose donc de son propre histogramme. Puis, dans chaque découpe rectangulaire de l'image faite par le quadrillage, nous regardons l'histogramme de la fenêtre et le comparons à nos 3 histogrammes définis précédemment. Ainsi, nous pouvons appliquer à chaque fenêtre un traitement différent, si c'est une pierre claire, foncée… etc. Pour terminer, il ne reste qu'à rassembler les segmentations produites sur chaque fenêtre. Autre aspect de la segmentation, cette dernière demande à l'utilisateur de fixer des paramètres. Si nous approfondissons les algorithmes, il y aura certainement d'autres paramètres à fixer. Il serait donc bon d'établir un outil qui génère automatiquement ces paramètres en vue d'obtenir un traitement optimum.

Au niveau de la structure même de notre outil, elle reste, malgré le travail effectué, relativement sommaire. L'écriture, à l'aide de la bibliothèque Tkinter, n'est pas poussée. Elle n'offre pas suffisamment de solutions pour le confort de l'utilisateur. Une meilleure gestion des boîtes de dialogue, comme nous l'avons vu, est envisagée (ex : regroupement des choix de paramètres en une seule fenêtre). De plus, des fonctionnalités comme un zoom, une *preview* du résultat… pourraient être

ajoutées au programme. Cependant, la bibliothèque Tkinter ne suffirait peut être pas pour gérer les évènements générés par ces fonctionnalités.

Concernant la dernière partie du projet, la classification supervisée est fonctionnelle. Nous obtenons un résultat néanmoins insatisfaisant : environ 4 pierres sur 5 sont mal classées (suivant la classification). Cela est certainement dû au fait que cette classification se repose sur des critères radiométriques fortement liés à la lumière ambiante. Il se propose alors à nous 3 solutions.

La première, nous ajoutons d'autres critères pour caractériser nos classes (comme des paramètres de forme).

La seconde serait d'analyser l'influence des échantillons utilisés dans la création des paramètres de classe pour en établir de meilleurs. La prise de photographie par HDR[10] serait une bonne solution pour éliminer l'effet de la lumière ambiante.

La troisième solution serait que, par une interface graphique, l'utilisateur puisse choisir une pierre de chaque catégorie présente sur son image. L'outil génèrerait ensuite automatiquement les caractéristiques de chaque classe en prenant comme échantillons de départ les pierres sélectionnées par l'utilisateur. De manière générale, nous pouvons améliorer les règles de décisions d'appartenance à chaque catégorie puisqu'elle se résume à une addition et une soustraction actuellement.

Nous avons fait le choix de programmer l'interface graphique de l'outil de segmentation en Python et d'utiliser celle du logiciel ArcMap pour les outils de classification. Malheureusement nous avons étudié et découvert la possibilité d'user de l'interface d'ArcMap pour nos outils en fin de projet. Nous n'avons pas eu le temps d'explorer cette nouvelle solution pour l'outil dans sa globalité. Il faudrait étudier la possibilité d'associer notre outil de segmentation à ArcMap (comme pour les outils de classification) : l'utilisation de la bibliothèque Tkinter ne serait alors plus nécessaire.

Ce projet présente deux points qui se confrontent : le premier oblige l'utilisateur à trouver et régler de nombreux paramètres pour pallier à la grande diversité de murs à traiter, tandis que le deuxième prône la simplicité d'utilisation pour l'utilisateur qui est un archéologue et non pas un féru de traitement d'image.

Pour conclure, il faut retenir la chose suivante : les aspects segmentation et classification de ce projet reste fortement à développer puisque nous n'avons pas eu le temps de les approfondir.
Néanmoins, il existe de nombreuses solutions possibles.

[10] High Dynamic Range : méthode de capture d'image permettant d'accentuer les contrastes naturels en limitant entre autre les effets de la lumière ambiante tels l'ombre

Bibliographie

CNES. 2013. Connected component segmentation module. *Wiki orfeo toolbox.* [En ligne] 14 Mars 2013. http://wiki.orfeo-toolbox.org/index.php/Connected_component_segmentation_module.

—. 2014. Mean Shift Segmentation Refactoring. *Wiki Orfeo Toolbox.* [En ligne] mars 2014. http://wiki.orfeo-toolbox.org/index.php/Mean_Shift_Segmentation_Refactoring#EDISON_Based_implementation.

—. 2014. Orfeo Toolbox. *Orfeo Toolbox.* [En ligne] 2014. http://orfeo-toolbox.org/otb/.

Efficient graph-based image segmentation, Felzenszwal. **Huttenlocher, P.F. and. 2004.** 2004, D.P. International Journal of Computer Vision.

ESRI. 2013. Resources Arcgis. [En ligne] 2013. http://resources.arcgis.com/fr/help/main/10.2/index.html#//001200000030000000.

Mean Shift Analysis and Applications. **D. Comaniciu and P. Meer. 1999.** Kerkyra, Greece : IEEE, 1999. International Conference Computer Vision (ICCV'99). pp. 1197-1203.

Meyer, S. Beucher and F. 1992. The Morphological Approach to Segmentation : The Watershed Transform. *Mathematical Morphology in Image Processing.* New York : Marcel Dekker, 1992, pp. 451-481.

OpenCV dev Team. 2014. Image Segmentation with Watershed Algorithm. *OpenCV 3.0.0 Documentation.* [En ligne] 21 Mai 2014. http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc/py_watershed/py_watershed.html.

Quick shift and kernel methods for mode seeking. **Soatto, A. and. 2008.** s.l. : Vedaldi, 2008. S. European Conference on Computer Vision.

Robert Fisher, Simon Perkins, Ashley Walker and Erik Wolfart. 2003. Gaussian Smoothing. *Image processing Learning resources.* [En ligne] 2003. http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm.

Scikit-image development team. 2014. Comparing edge-based segmentation and region-based segmentation. *Scikit-image, Image processing in Python.* [En ligne] 2014. http://scikit-image.org/docs/0.9.x/auto_examples/applications/plot_coins_segmentation.html#example-applications-plot-coins-segmentation-py.

—. 2014. Comparison of segmentation and superpixel algorithms. *Scikit-image, image processing in python.* [En ligne] avril 2014. http://scikit-image.org/docs/dev/auto_examples/plot_segmentations.html#example-plot-segmentations-py.

—. 2014. Random walker segmentation. *Scikit-image, image processing in python.* [En ligne] avril 2014. http://scikit-image.org/docs/dev/auto_examples/plot_random_walker_segmentation.html#example-plot-random-walker-segmentation-py.

Wikipedia Foundation. 2013. Filtre de Sobel. *Wikipedia, L'encyclopédie libre.* [En ligne] 1 Avril 2013. http://fr.wikipedia.org/wiki/Filtre_de_Sobel.

—. 2013. Ligne de partage des eaux (segmentation). *Wikipedia, L'encyclopédie libre.* [En ligne] 16 Septembre 2013. http://fr.wikipedia.org/wiki/Ligne_de_partage_des_eaux_%28segmentation%29.

Y. Ukrainitz and B. Sarel. 2003. Segmentation. [En ligne] 2003. http://cs.nyu.edu/~fergus/teaching/vision/22_23_Segmentation.pdf.

Table des figures

Table des annexes

Annexe n°1 : Paramètres d'entrée et sortie des algorithmes de segmentation

Paramètre de sortie commun à tous les algorithmes :

- *Output vector file/labeled image* : fichier de sortie (vecteur ou raster).

Segmentation *Mean-Shift*

Paramètres d'entrée :

- *Input Image* : image à traiter,
- *Spatial radius* : rayon spatial au voisinage du pixel concerné (lié à la résolution spatiale de l'analyse) → affecte le lissage et la connectivité des segments,
- *Range radius* : rayon (exprimé en unité radiométrique) dans l'espace multispectral → affecte le nombre de segments (doit être maintenu bas si contraste faible),
- *Mode convergence threshold* : algorithme itératif qui s'arrête si le vecteur *Mean-Shift* est inférieur au seuil ou si le nombre d'itérations atteint un nombre maximum d'itérations - Paramètre exclusif à la méthode de *Mean-Shift*,
- *Maximum number of iterations* : nombre maximum d'itérations,
- *Scale factor* : facteur d'échelle de l'image d'entrée avant le traitement. C'est utile pour celles avec des gammes de valeurs étroites spectrales (comme [0,1]). - Paramètre exclusif à la méthode « Edison »,
- *Minimum region size* : Taille minimale d'une région (en pixel) dans la segmentation. Les petites régions seront fusionnées au voisin de radiométrie la plus proche. Si définie à 0, aucune taille minimale n'est demandée. (Doit être choisi en fonction de la taille des régions « nuisibles »).

Segmentation *Connected components*

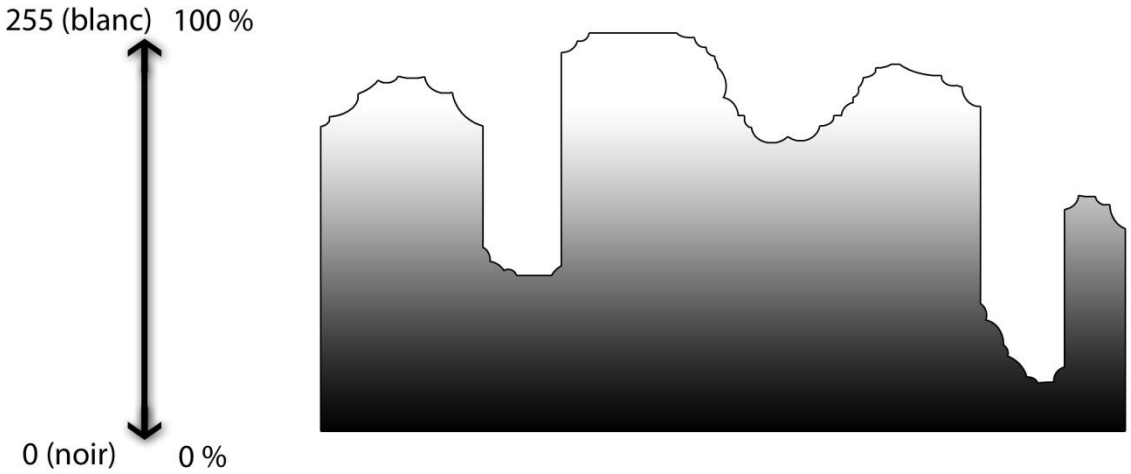
Paramètres d'entrée :

- *Input Image* : image à traiter,
- Condition : l'utilisateur définit la condition de connexion avec une expression mathématique. Les variables disponibles sont p(i)b(i) qui signifie le canal (b = band) i du pixel i, intensity_p(i) qui signifie l'intensité du pixel i et la distance euclidienne entre deux pixels adjacent. Les fonctions utilisables sont référencées, elles sont pour la plupart des fonctions mathématiques comme le cosinus ou le logarithme.

Segmentation *Watershed*

Paramètres d'entrée :

- *Input Image* : image à traiter,
- *Depth threshold* : l'unité du seuil de profondeur est un pourcentage (entre 0 et 1) du maximum de profondeur dans l'image (voir schéma explicatif ci-dessous de cette représentation topographique d'un pixel en fonction de sa valeur de niveau de gris),



- *Flood level* : niveau de la crue/d'inondation pour générer l'arborescence de fusion à partir de la segmentation initiale (entre 0 et 1).

Segmentation *Morphological profiles based*

Paramètres d'entrée :

- *Input Image* : image à traiter,
- *Profile size* : taille des profils,
- *Initial radius* : rayon initial de l'élément structurant (en pixels),
- *Radius Step.* : pas du rayon le long du profil (en pixels),
- *Threshold of the final decision rule* : les valeurs des profils sous le seuil seront ignorées.

Segmentation *Edge based*

Paramètres d'entrée :

- Niveau de seuil minimum (> ou <) sur la base de l'histogramme des niveaux de gris de l'image.
- Taille minimale des régions.
- Paramètres de l'algorithme de Canny :
    - (Sigma) Déviation standard du filtre *Gauss Smoothing* : plus le sigma sera grand et plus le lissage par le filtre de Gauss sera important (Robert Fisher, 2003).
    - Seuil haut et bas du seuillage par hysteresis : Tout d'abord, on étiquette les pixels (*edge pixel* ou *non-edge pixel*) qui sont au-dessus du seuil haut, puis on étiquette les pixels qui sont en-dessus du seuil bas et qui sont *8-connected* à un pixel déjà étiqueté comme contour.

Segmentation *Region based*

Paramètre d'entrée :

- Paramétrage des seuils haut et bas pour fixer les marqueurs de fond et de régions d'intérêt.

Segmentation *Random walker*

Paramètres d'entrée :

- Les paramètres concernant les marqueurs « région d'intérêt » et « d'arrière-plan »,
- Les paramètres de la segmentation *Random walker* dont *Beta* : plus ce chiffre sera grand et plus la diffusion d'un pixel à un autre sera difficile.

Segmentation *Felzenszwalb*

Paramètres d'entrée :

- *scale* : Plus l'échelle est grande et plus les segments seront grands (mais moins nombreux),
- *sigma* : Paramètre utilisé dans le « lissage » de l'image par le filtre gaussien avant l'application de l'algorithme *Felzenszwalb*,
- *min-size* : taille minimale des composantes après traitement.

Annexe n°2 : Brèves caractéristiques des 7 catégories de pierre

Nom	Couleur	Texture	Résistance	Dimensions
Brique	Rouge à orange	Lisse	Friable	Quelques cm d'épaisseur, longue
Calcaire de bernay	Blanc jaunâtre	lisse	Moyenne	Variable selon plusieurs dizaines de cm
Granit	Gris bleuté	Rugueuse	Dur	Gros bloc de 50cm et plus
Grès cénomanien	Orange-jaune	Rugueuse et sableuse	Friable	Forme irrégulière
Grès éocène	Gris	Lisse ou striée	Dur	Rectangulaire, 20 à 40cm de long
Grès roussard	Marron à rougeâtre	Rugueuse et sableuse	Moyenne	Moellons tronconiques, diamètre 10-15cm
Tuffeau	Blanc	Crayeuse	Friable	Moellons tronconiques, diamètre 10-15cm

Annexe n°3 : Statistiques zonales de briques, canaux RGB

Canal rouge

ID	MIN	MAX	MEAN	STD
0	50	190	112	30
1	65	190	125	25
2	62	186	125	28
3	76	181	128	24
4	58	172	121	27
5	71	179	133	32
6	64	160	98	22
MOY	64	180	120	27

Canal vert

ID	MIN	MAX	MEAN	STD
0	31	160	85	24
1	46	164	96	21
2	49	161	98	22
3	50	151	92	19
4	36	141	87	20
5	50	157	103	28
6	47	133	77	20
MOY	44	152	91	22

Canal bleu

ID	MIN	MAX	MEAN	STD
0	22	141	70	22
1	36	147	84	19
2	26	158	83	25
3	33	135	76	19
4	21	119	71	19
5	42	144	89	26
6	36	114	64	19
MOY	31	137	77	21

Annexe n°4 : Statistiques zonales de calcaires de bernay, canaux RGB

Canal rouge

ID	MIN	MAX	MEAN	STD
0	123	177	147	8
1	110	174	138	9
2	122	169	143	7
3	119	160	140	7
4	112	173	141	7
5	104	165	148	6
6	120	161	143	7
7	115	176	136	8
8	127	178	142	6
9	122	163	141	7
MOY	117	170	142	7

Canal vert

ID	MIN	MAX	MEAN	STD
0	123	177	148	8
1	116	177	139	9
2	123	170	144	6
3	121	162	141	6
4	120	174	143	6
5	103	166	149	7
6	124	162	146	6
7	117	168	138	7
8	128	181	143	6
9	122	162	143	7
MOY	120	170	143	7

Canal bleu

ID	MIN	MAX	MEAN	STD
0	130	183	155	8
1	119	184	143	9
2	129	173	150	6
3	127	168	148	6
4	121	179	149	7
5	104	171	156	7
6	129	169	152	6
7	123	163	143	7
8	135	181	150	5
9	128	170	149	7
MOY	125	174	150	7

Annexe n°5 : Statistiques zonales de granits, canaux RGB

Canal rouge

ID	MIN	MAX	MEAN	STD
0	13	154	66	20
1	14	177	63	15
MOY			65	18

Canal vert

ID	MIN	MAX	MEAN	STD
0	20	150	74	17
1	12	184	63	17
MOY			69	17

Canal bleu

ID	MIN	MAX	MEAN	STD
0	8	146	54	25
1	15	229	66	18
MOY			60	22

Annexe n°6 : Statistiques zonales de grès cénomaniens, canaux RGB

Canal rouge

ID	MIN	MAX	MEAN	STD
0	78	191	115	16
1	54	219	150	24
2	64	202	132	21
3	76	230	136	24
4	91	195	149	16
5	80	165	122	16
6	81	175	134	17
7	71	188	132	18
8	78	173	129	19
9	82	175	131	17
MOY	76	191	133	19

Canal vert

ID	MIN	MAX	MEAN	STD
0	71	181	104	15
1	47	214	138	24
2	59	192	121	20
3	69	233	125	26
4	79	173	133	15
5	64	160	112	16
6	76	167	122	15
7	58	180	119	18
8	69	164	120	21
9	76	163	114	17
MOY	67	183	121	19

Canal bleu

ID	MIN	MAX	MEAN	STD
0	55	174	95	15
1	47	217	120	24
2	49	183	107	20
3	47	242	112	29
4	59	150	113	15
5	48	159	96	17
6	66	152	108	15
7	35	189	104	20
8	57	176	112	26
9	58	147	102	18
MOY	52	179	107	20

Annexe n°7 : Statistiques zonales de grès éocènes, canaux RGB

Canal rouge

ID	MIN	MAX	MEAN	STD
0	35	172	109	21
1	40	164	104	20
2	21	188	89	22
3	57	156	109	15
MOY	38	170	103	19

Canal vert

ID	MIN	MAX	MEAN	STD
0	41	183	109	21
1	37	171	108	20
2	30	202	89	22
3	57	159	110	15
MOY	41	179	104	19

Canal bleu

ID	MIN	MAX	MEAN	STD
0	39	202	114	23
1	38	182	116	23
2	23	226	91	22
3	59	171	118	16
MOY	40	195	110	21

Annexe n°8 : Statistiques zonales de grès roussards, canaux RGB

Canal rouge

ID	MIN	MAX	MEAN	STD
0	51	121	76	12
1	56	125	84	10
2	58	105	84	8
3	55	103	79	7
4	54	103	75	8
5	63	121	86	8
6	51	102	75	9
7	50	107	82	12
8	49	102	75	7
9	53	103	80	9
10	45	100	76	9
MOY	53	108	79	9

Canal vert

ID	MIN	MAX	MEAN	STD
0	44	108	69	11
1	44	116	74	10
2	44	86	65	7
3	47	95	73	7
4	46	92	65	8
5	52	115	76	9
6	43	87	65	9
7	41	93	70	11
8	36	91	62	8
9	44	92	69	8
10	39	90	66	8
MOY	44	97	69	9

Canal bleu

ID	MIN	MAX	MEAN	STD
0	37	98	64	12
1	36	105	66	11
2	35	82	57	7
3	44	101	73	8
4	39	87	60	8
5	41	103	66	9
6	39	84	64	8
7	37	87	64	10
8	29	88	56	9
9	37	93	64	10
10	37	87	64	8
MOY	37	92	63	9

Annexe n°9 : Statistiques zonales de tuffeaux, canaux RGB

Canal rouge

ID	MIN	MAX	MEAN	STD
0	65	189	133	34
1	76	173	139	19
2	78	187	126	21
3	64	160	113	16
4	64	161	109	17
5	68	168	124	17
6	49	195	132	30
7	65	168	110	24
8	68	164	115	21
MOY	66	174	122	22

Canal vert

ID	MIN	MAX	MEAN	STD
0	61	198	134	38
1	69	173	134	22
2	73	198	125	24
3	58	158	108	17
4	59	163	105	18
5	68	166	121	18
6	44	203	129	33
7	62	167	105	24
8	65	175	115	22
MOY	62	178	120	24

Canal bleu

ID	MIN	MAX	MEAN	STD
0	61	203	136	38
1	65	167	126	22
2	68	196	119	25
3	55	155	101	17
4	48	167	104	20
5	66	169	118	19
6	39	200	124	34
7	51	168	101	26
8	61	181	116	25
MOY	57	178	116	25

Annexe n°10 : Statistiques zonales de briques, canaux HSV

Canal Teinte

FID	MIN	MAX	MEAN	STD
0	7	37	15	4
1	0	53	13	5
2	0	252	16	15
3	8	24	13	3
4	7	23	13	3
5	6	21	13	3
6	10	23	16	2
	5	62	14	5

Canal Saturation

FID	MIN	MAX	MEAN	STD
0	39	147	97	19
1	27	125	83	15
2	26	143	86	20
3	64	145	105	14
4	71	158	107	13
5	46	109	85	13
6	57	115	89	13
	47	135	93	15

Canal Luminosité

FID	MIN	MAX	MEAN	STD
0	49	189	112	30
1	65	190	125	25
2	61	185	125	28
3	76	183	128	24
4	58	173	121	27
5	70	180	133	32
6	64	160	98	22
	63	180	120	27

Annexe n°11 : Statistiques zonales de calcaires de bernay, canaux HSV

Canal Teinte

ID	MIN	MAX	MEAN	STD
0	146	244	163	9
1	0	233	154	36
2	0	198	163	12
3	136	198	163	8
4	127	205	162	8
5	0	252	163	15
6	142	200	161	8
7	0	233	162	16
8	0	244	163	18
9	145	233	164	8
MOY	70	224	162	14

Canal Saturation

ID	MIN	MAX	MEAN	STD
0	3	24	14	3
1	0	19	9	4
2	0	21	12	3
3	3	25	13	3
4	3	25	13	3
5	3	38	12	3
6	5	23	14	3
7	0	25	12	3
8	0	24	14	3
9	3	21	12	3
MOY	2	25	13	3

Canal Luminosité

ID	MIN	MAX	MEAN	STD
0	129	185	155	8
1	120	182	143	9
2	128	175	150	6
3	128	168	148	6
4	122	179	149	7
5	110	171	156	7
6	129	170	152	6
7	121	175	143	7
8	136	182	150	5
9	129	170	149	7
MOY	125	176	150	7

Annexe n°12 : Statistiques zonales de granits, canaux HSV

Canal Teinte

ID	MIN	MAX	MEAN	STD
0	0	244	62	20
1	0	250	139	68
MOY			100	44

Canal Saturation

ID	MIN	MAX	MEAN	STD
0	0	231	77	51
1	0	119	20	12
MOY			49	31

Canal Luminosité

ID	MIN	MAX	MEAN	STD
0	20	154	74	17
1	15	229	67	17
MOY			70	17

Annexe n°13 : Statistiques zonales de grès cénomaniens, canaux HSV

Canal Teinte

ID	MIN	MAX	MEAN	STD
0	0	252	30	44
1	0	250	29	27
2	0	252	27	26
3	0	246	32	34
4	0	246	27	29
5	0	250	30	27
6	0	247	26	20
7	0	252	29	35
8	0	244	39	47
9	0	249	19	15
MOY	0	249	29	30

Canal Saturation

ID	MIN	MAX	MEAN	STD
0	0	110	45	19
1	3	119	53	20
2	0	100	47	18
3	0	114	47	24
4	4	106	61	18
5	3	124	54	22
6	4	105	50	17
7	0	149	55	24
8	0	91	41	21
9	18	100	58	15
MOY	3	112	51	20

Canal Luminosité

ID	MIN	MAX	MEAN	STD
0	79	191	115	16
1	58	219	150	24
2	63	202	132	21
3	77	244	136	24
4	88	195	149	16
5	81	164	122	16
6	84	175	134	17
7	71	190	132	18
8	78	178	131	21
9	82	176	131	17
MOY	76	193	133	19

Annexe n°14 : Statistiques zonales de grès éocènes, canaux HSV

Canal Teinte

ID	MIN	MAX	MEAN	STD
0	0	250	168	42
1	0	244	153	20
2	0	250	134	69
3	0	244	167	16
MOY	0	247	156	37

Canal Saturation

ID	MIN	MAX	MEAN	STD
0	0	62	16	7
1	0	65	26	8
2	0	106	18	11
3	0	36	20	6
MOY	0	67	20	8

Canal Luminosité

ID	MIN	MAX	MEAN	STD
0	43	201	115	22
1	40	182	116	22
2	32	223	94	22
3	59	170	118	15
MOY	44	194	111	21

Annexe n°15 : Statistiques zonales de grès roussards, canaux HSV

Canal Teinte

ID	MIN	MAX	MEAN	STD
0	0	250	24	41
1	7	29	19	3
2	5	21	12	3
3	0	250	104	116
4	2	22	13	4
5	11	31	20	3
6	0	250	80	112
7	0	21	14	4
8	2	22	12	3
9	0	250	34	67
10	0	19	8	4
MOY	2	106	31	33

Canal Saturation

ID	MIN	MAX	MEAN	STD
0	15	91	41	14
1	28	104	56	12
2	39	116	80	16
3	6	55	21	8
4	30	79	49	8
5	34	94	60	11
6	23	86	40	9
7	30	79	56	9
8	34	122	63	15
9	16	86	50	13
10	24	58	39	5
MOY	25	88	50	11

Canal Luminosité

ID	MIN	MAX	MEAN	STD
0	50	121	76	12
1	57	125	84	10
2	58	106	84	8
3	55	102	79	7
4	55	102	74	8
5	62	122	86	8
6	50	102	75	9
7	50	108	82	12
8	49	102	74	7
9	53	104	80	9
10	45	100	76	9
MOY	53	109	79	9

Annexe n°16 : Statistiques zonales de tuffeaux, canaux HSV

Canal rouge

ID	MIN	MAX	MEAN	STD
0	0	250	120	67
1	0	250	31	32
2	0	250	58	52
3	0	250	29	29
4	0	250	95	97
5	0	244	55	65
6	0	250	54	59
7	0	252	44	59
8	0	244	89	72
MOY	0	249	64	59

Canal vert

ID	MIN	MAX	MEAN	STD
0	0	72	15	9
1	0	58	24	13
2	0	57	21	11
3	0	62	27	10
4	0	65	16	12
5	0	59	15	10
6	0	70	21	14
7	0	100	23	15
8	0	59	12	9
MOY	0	67	19	11

Canal bleu

ID	MIN	MAX	MEAN	STD
0	64	201	138	37
1	77	173	139	20
2	79	198	128	23
3	64	161	113	16
4	64	167	110	17
5	70	169	124	17
6	48	202	133	31
7	64	170	110	24
8	68	181	118	24
MOY	66	180	124	23

CAPRA — RECHERCHE & VALORISATION ARCHÉOLOGIQUES

Développement d'un outil de segmentation et de classification semi-automatique de pierres

le cnam — École supérieure des géomètres et topographes — esgt

Contexte et objectifs

Contexte : **l'archéologie du bâti nécessite l'étude des étapes de construction et de reconstruction des murs/façades**

Objectif : **réduire le temps de traitement des archéologues**

Moyen : **création d'un outil semi-automatique de détection et de classification des contours de pierres**

Exemple de cas traité par le CAPRA : Image de Pavés de sol superposée aux contours de ces derniers

Image servant de test à notre outil de segmentation et de classification automatique : Mur de la cathédrale du MANS

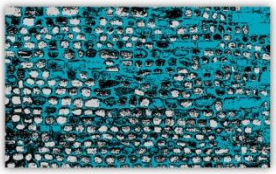
Segmentation

2 bibliohtèques libres de traitement d'image étudiées :
Orfeo Toolbox et Scikit-Image

1 algorithme retenu : Region based segmentation (~watershed)

+ Cet algorithme utilise la méthode des bassins versants (basée sur du filtrage morphologique). Les pixels se voient attribués une altitude topographique en fonction de leur valeur de niveau de gris :

+ 2 paramètres d'entrée :
  - Seuil bas : marqueurs d'arrière-plan
  - Seuil haut : marqueurs de régions d'intérêt



Classification des pixels (seuil bas à 80 -> marqueurs bleus, seuil haut à 140 -> marqueurs blanc)

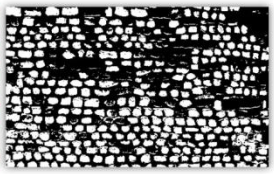


Image binaire résultant de la segmentation

Programmation de l'interface graphique accueillant l'algorithme :
+ Utilisation du langage de programmation Python,
+ Sélection de 3 bibliothèques de fonctionnalités (Tkinter : interface graphique, Scikit-image : algorithme de traitement d'image, Pillow : manipulation des images).

Extrait de la fenêtre principale de l'outil

Classification supervisée

Prétraitement :
+ Transformation de l'image binaire en vecteur (utilisation de l'outil «raster vers polygone» du logiciel ArcMap)
+ Modification de ce vecteur à partir de la barre d'outils «Éditeur» d'ArcMap

Choix des classes à la base de la classification supervisée
8 classes -> 7 catégories de pierres localement (Sarthe) utilisées :
+ brique (1),
+ calcaire de Bernay (2),
+ granit (3),
+ grès cénomanien (4),
+ grès éocène (5),
+ grès roussard (6),
+ tuffeau (7).

-> une 8ème catégorie «Autre» pour combler les lacunes.

Méthodologie suivie :
+ Détermination de paramètres propres à chaque classe,
+ Détermination des règles de décision (d'appartenance à chaque classe) suivant les différents paramètres,
+ Application du traitement à notre image segmentée :
  -> Détermination des classes de chacune des pierres de l'image.

Résultats :
+ Deux outils implémentés dans ArcMap permettant de générer automatiquement les statistiques des images (pour les canaux radiométriques) puis de classifier dans un tableau ArcMap les pierres.
+ Traitement produit à partir :
Premièrement des canaux Rouge, Vert et bleu de l'image (RVB),
Deuxièmement des canaux Teinte, saturation et luminosité (TLS).

FID	Shape *	OBJECTID	Id	gridcode	Shape_Leng	Shape_Area	Valeur	Valeur2	classe	classif	classhsv
197	Polygone	2346	204	0	0.38886	0.008508	255	130	2	2	5
198	Polygone	2347	205	0	0.615532	0.018277	255	130	2	2	4
199	Polygone	2348	206	0	0.279086	0.005057	255	130	6	1	1
200	Polygone	2349	207	0	0.56095	0.016323	255	130	2	4	5
201	Polygone	2350	208	0	0.263489	0.00496	255	130	2	2	2
202	Polygone	2351	209	0	0.358908	0.008639	255	130	6	1	1
203	Polygone	2352	210	0	0.383355	0.010053	255	130	6	3	3
204	Polygone	2353	211	0	0.344584	0.008923	255	130	6	1	8
205	Polygone	2354	212	0	0.352468	0.007836	255	130	6	6	3
206	Polygone	2355	213	0	0.26227	0.005176	255	130	6	1	1
207	Polygone	2356	214	0	0.311195	0.006796	255	130	6	1	3
208	Polygone	2357	215	0	0.327936	0.007763	255	130	6	1	6
209	Polygone	2358	216	0	0.282763	0.004068	255	130	6	1	6

Extrait du tableau ArcMap résultant de la classification

Colonne «classhsv» comporte les résultats de classification à partir des canaux TLS, colonne «classif» comporte les résultats de celle produite par les canaux RVB, la colonne «classe» représente la vérité terrain.

État et perspectives

Algorithme de segmentation limité : *+ Utilisation de seuil sur les valeurs d'intensité des pixels* *-> Empêche la détection simultanée des pixels clairs et foncés.* *+ Paramètres très variables suivant l'échelle et le type d'image*	*+ Approfondir les comportements des autres algorithmes suivant le type de pierre* *+ Combinaison de plusieurs segmentations ?* *+ Élaboration automatique des paramètres pour un traitement optimal*
Interface fonctionnelle	*+ Ajouter de nouvelles fonctionnalités et améliorer l'ergonomie.*
Résultats classification faibles (461 pierres au total) : *+ 20,82 % des pierres bien classées à partir des paramètres TLS,* *+ 14,32 % des pierres bien classées à partir des paramètres RVB.*	*+ Etablir d'autres paramètres (de forme, HSV...) pour caractériser les catégories de pierres.* *+ Améliorer les règles de décision de classification.* *+ Analyser l'influence des échantillons utilisés dans la création des paramètres de classe.*

Titre : « Développement d'un outil de segmentation et de classification semi-automatique de pierres »

Auteur : Hugueny Pierre-Alban

Résumé :

Le travail des archéologues pour étudier les étapes de construction et de reconstruction des façades est aujourd'hui fastidieux entre autre parce qu'ils doivent, à la main, dessiner et qualifier les pierres des murs. Afin de résoudre ce problème, nous avons développé, en collaboration avec le CAPRA, un ensemble d'outils. Une première partie, liée à la détection des contours, est de programmer, dans le langage Python, une interface graphique et d'y implémenter notre algorithme de segmentation. La seconde partie, concernant la classification supervisée du résultat automatique, utilise une succession de deux outils intégrés à l'interface d'ArcMap pour générer automatiquement la classification de ces pierres à partir de l'image segmentée. Les outils se voulant semi-automatiques, l'utilisateur peut modifier à chaque fois les résultats produits. Cependant, des limitations sont bien présentes et particulièrement concernant l'algorithme de segmentation qui n'est pas complètement ajusté à la détection des pierres sur des murs dégradés par le temps et la nature, ou dont les joints ne sont pas propres.

Mots-clés : Traitement numérique des images, segmentation, détection de contour, pierres, programmation Python, interface graphique, classification supervisée

Abstract :

Today, the work of archaeologists to study the steps of construction and reconstruction of facades is boring because they have to draw and qualify stones of walls. In order to solve this problem, we developed a set of tools. The first part, related to edge detection, is to program, using the informatic language Python, a graphical user interface and implementing our segmentation algorithm in it. The second part, concerning the supervised classification of our automatic result, uses a succession of two tools integrated in the software ArcMap to generate automatically the classification of these stones from the segmented image. Tools aiming to be semi-automatic, the user can modify every time the produced results. However, limitations are always present and particularly concerning the segmentation algorithm which is not completely adjusted to the detection of stones on degraded walls by weather or nature or whose joints are not clean.

Key-words : Digital image processing, segmentation, edge detection, stones, Python programmation, Graphical interface, supervised classification