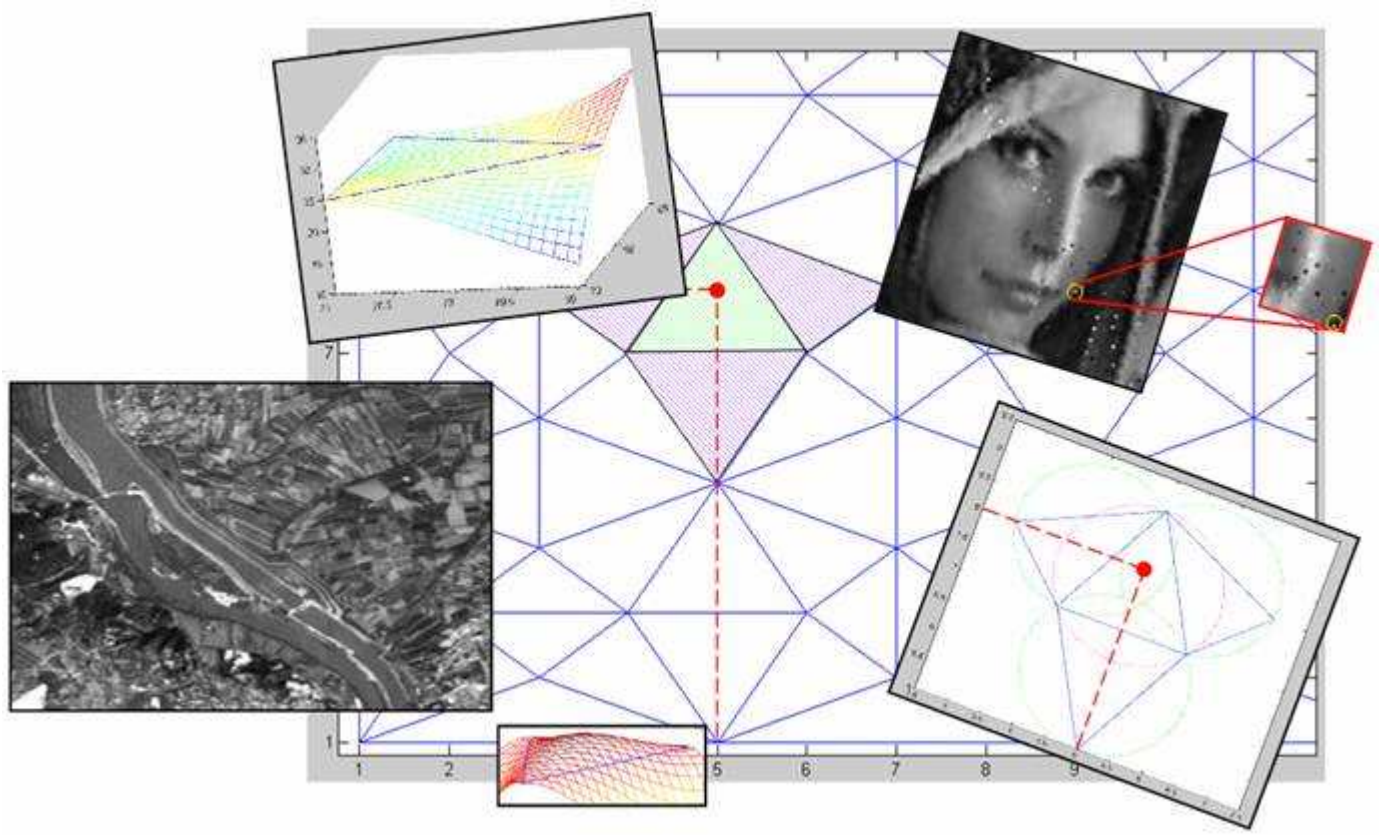
Mastère de Recherche “Mathématiques, vision, apprentissage”

La triangulation de Delaunay: application au problème de la superresolution

Cours de traitement d’images sous la direction d’Henri Maître
Décembre 2004

Auteurs :
Cécilia Damon ceciliadamon@hotmail.com
Francisco Sánchez Vega sanchez@enst.fr



L'objectif de ce projet est de produire une image de haute résolution à partir de plusieurs images de basses résolutions en utilisant la triangulation de Delaunay qui est un type de partitionnement du support d'une image.

Dans une première partie, on décrira donc la triangulation de Delaunay comme dual du diagramme de Voronoï ainsi que leurs principales propriétés mathématiques et leurs principales applications pratiques. On proposera également une applet java sur la triangulation de Delaunay et le pavage de Voronoï afin d’illustrer ces deux concepts.

Dans une deuxième partie, on s’intéressera tout particulièrement à une application au problème de la superresolution et on présentera des travaux et des simulations que nous avons réalisées en Matlab.

Sommaire

Généralités et propriétés mathématiques

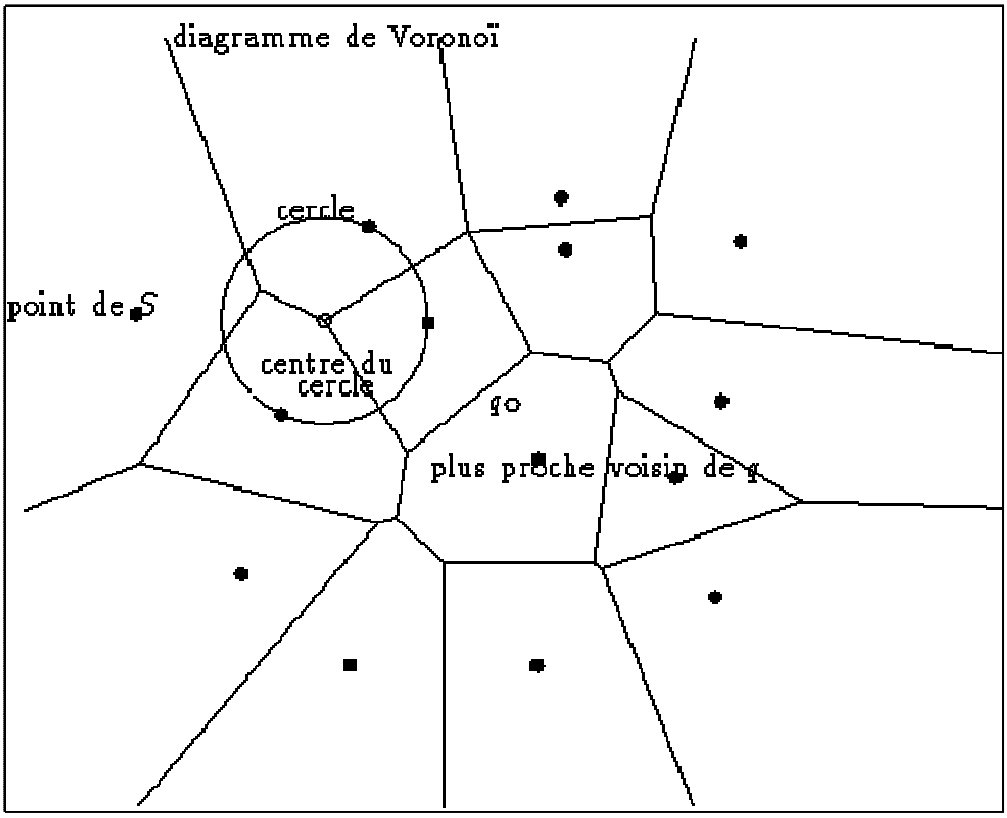
a) Le diagramme de Voronoï

Voronï était un mathématicien russe des années 1900. Au début du XXème siècle, il invente une partition d'un plan E composé d'objets en zones où chaque zone est définie comme l'ensemble des points les plus proches d'un objet :

zone(objet(i))={ x élément de E tel que d(x,objet(i))<=d(x,objet(j)) quelque soit j différent de i}

Cette partition d'un plan est appelé Diagramme de Voronoï. Chaque zone de Voronoï est un polygone convexe.

Figure: Diagramme de Voronoï



Les diagrammes de Voronoï permettent de représenter des relations de distance entre objets : ils sont souvent utilisés pour modéliser des réseaux comme les cristaux ou les grandes structures de l'univers.

On les trouve également dans la nature, par exemple sur la carapace d'une tortue ou sur le cou d'une girafe réticulée.

Le concept a d'abord été utilisé dans les travaux de Descartes en 1644 pour montrer la position des astres dans le système solaire et ses environs. Puis généralisé par Voronoï, il a beaucoup servi à la météorologie et il est actuellement largement employé dans des nombreux domaines scientifiques différents comme par exemple les mathématiques, la biologie, l'informatique, la cartographie ou la physiologie.

b) Triangulation de Delaunay

Delaunay était un autre mathématicien russe qui a étendu les travaux de Voronoï. La triangulation de Delaunay est un type de partitionnement d'un ensemble de points E positionnés dans un plan formé de triangles dont les sommets sont des objets, et qui à eux tous constituent une partition de l'enveloppe convexe de ces objets.

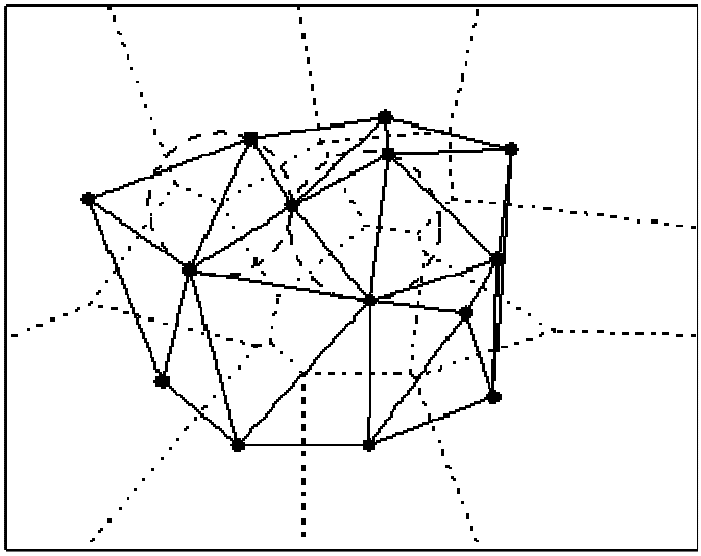
La triangulation de Delaunay a deux propriétés principales :

Le "critère du cercle": un triangle de Delaunay est un triangle qui a comme sommet trois objets, et tel que son cercle circonscrit n'ait en son intérieur aucun autre objet.

La triangulation de Delaunay est parmi toutes les triangulations de E celle qui maximise l'angle minimum de tous les triangles.

Du point de vue de la théorie des graphes, la triangulation de Delaunay est définie comme le diagramme dual du diagramme de Voronoï de telle façon que deux objets sont liés par un arc dans la triangulation de Delaunay s'ils appartiennent à des régions de Voronoï adjacentes.

Figure: Triangulation de Delaunay (trait pleins) et diagramme de Voronoï (pointillé)



Cette structure est très utile, par exemple, en mécanique, car c'est celle qui permet de mailler les objets de la manière la plus efficace, en minimisant les aires au carré des triangles. Elle est très utilisé aussi dans beaucoup d'autres domaines, dont celui qui nous occupe ici: le traitement d'image, et plus concrètement l'application au problème de la superresolution.

Algorithme de superresolution

Afin de montrer l'utilité de la triangulation de Delaunay dans un cas pratique de traitement d'image, on a décidé d'étudier et d'implémenter en Matlab les idées décrites dans l'article de référence ***«High Resolution Image Formation From Low Resolution Frames Using Delaunay Triangulation»*** de S. Lettrattanapanich et N. K. Bose, publié en *IEEE Transactions on Image Processing*, vol. 11, no. 12, Décembre 2002.

Ces auteurs ont proposé une méthode utilisant la triangulation de Delaunay pour résoudre le problème de la superresolution d'images à partir d'imagettes de plus basse résolution.

L'article décrit par ailleurs toute la problématique autour du sujet de la superresolution, en particulier le filtrage adapté et des techniques de *deblurring* que l'on ignorera ici pour se concentrer sur l'utilisation de la triangulation de Delaunay.

On va d'abord expliquer les différentes étapes de l'implémentation de l'algorithme que nous avons réalisé en Matlab, ainsi que les difficultés rencontrées.

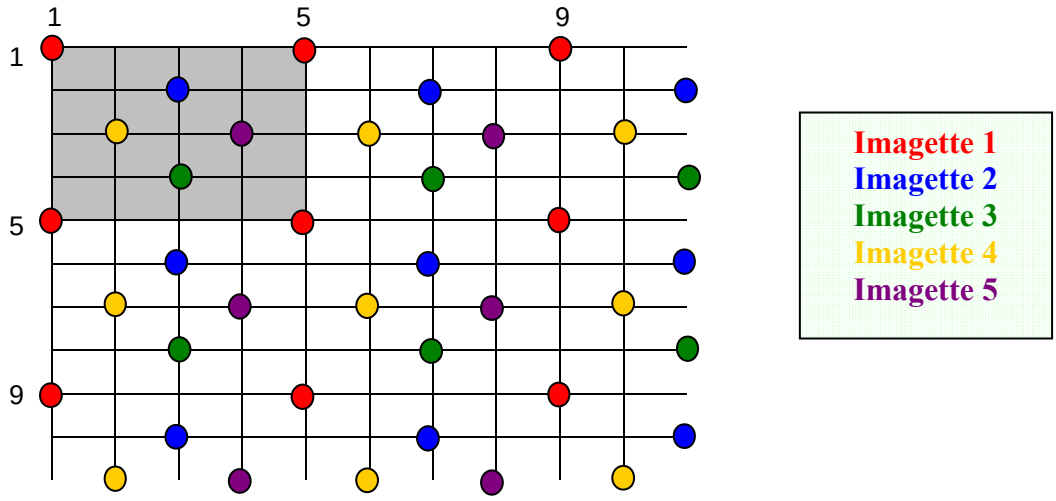
Étape 1 : Obtention d'un ensemble d'imagettes d'exemple

Dans le cas idéal, on aurait utilisé des imagettes de basse résolution représentant une même scène et prises à l'aide d'une camera ou d'un appareil photo, avec un certain déplacement relatif entre elles. Pour notre algorithme, nous avons crée les imagettes de basse résolution à partir d'une image de haute résolution par sous échantillonnage.

On peut travailler de cette manière parce que, de même que dans l'article de référence, nous supposons connu le déplacement relatif entre les imagettes.

Pour nos premières simulations, on a décidé de sous échantillonner 5 fois une image de « haute » résolution (100x100) avec un « pas d'échantillonnage» de 4 pixels. On a donc obtenu 5 imagettes de résolution 25x25 pixels en récupérant les pixels espacés de quatre en quatre par ligne et par colonne. Ces 5 imagettes sont différentes car leurs premiers pixels récupérés sur l'image de haute résolution ont tous des coordonnées différentes et par construction, tous les autres pixels aussi.

Pour la création des 5 imagettes, on a suivi le schéma suivant :



Le code Matlab pour l'obtention de chacune de ces imagettes est de la forme :

> lr_*id* =zeros(100);
> lr(n:4:100,m:4:100)=image_HR(n:4:100,m:4:100);

(où _id est le numéro d'imagette, et n et m son les coordonnées d'origine prises comme référence pour établir le déplacement relatif entre les différentes imagettes.)

Ce code génère des matrices « creuses » de 100x100 avec des bonnes valeurs de niveau de gris pour les pixels correspondant aux schémas d'échantillonnage choisis pour chaque cas, et des zéros pour le reste (ce qui sera très utile pour l'étape 2).

Tout d'abord on a considéré un morceau central de l'image de Lena pour nos expériences :

Les images en basse résolution ainsi obtenues sont les suivantes:

***Étape 2:* triangulation de Delaunay en 2D**

Une fois que l'on a obtenu les différentes imagettes, sous la forme de matrices «creuses» 100x100, on combine toutes les imagettes de basse résolution en sommant les 5 matrices obtenues pour la reconstruction de l'image de haute résolution :

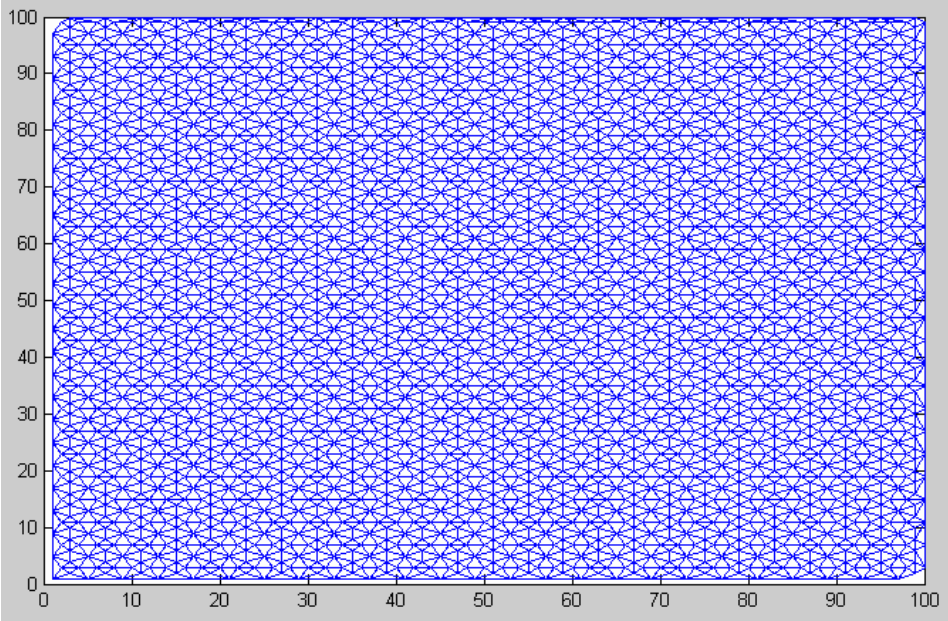
> **lt = lr1 + lr2 + lr3 + lr4 + lr5**

Cette opération équivaut à placer les points d'échantillonnage des 5 matrices sur une même matrice 100x100. Sur cette nouvelle matrice qui regroupe toute l'information récupérée, on réalisera la triangulation de Delaunay en 2D.

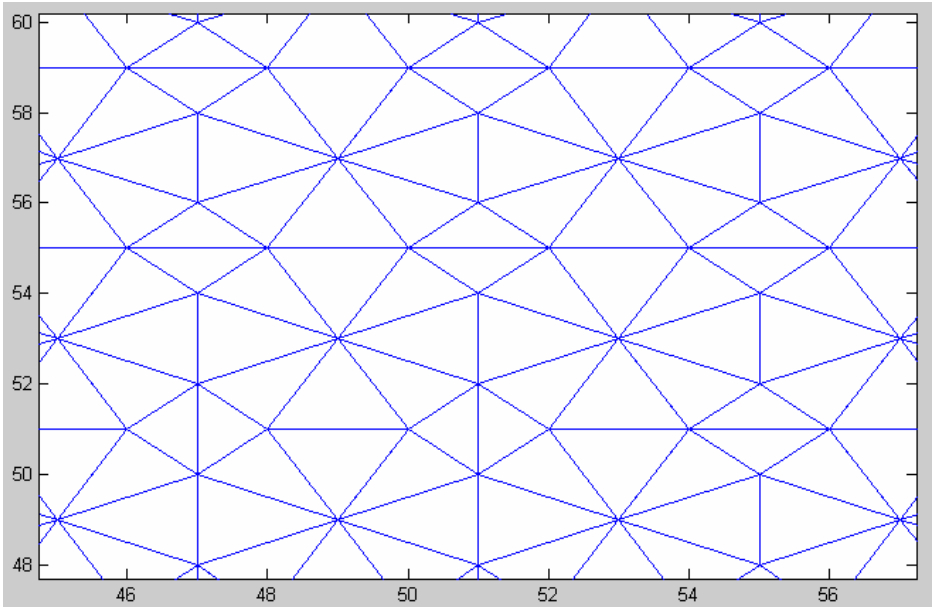
Ainsi, après avoir généré une grille de points appropriés avec la commande «*meshgrid*», le calcul de la triangulation de Delaunay en 2D est obtenu de manière automatique grâce à la fonction «delaunay» prédéfinie en Matlab,

```
> [X_id,Y_id]=meshgrid(n:4:100, m:4:100);      % pour _id=1:5
  X_id=X(:)';  Y_id=Y(:)';                     % pour _id=1:5
  X = [X1 X2 X3 X4 X5]
  Y = [Y1 Y2 Y3 Y4 Y5]
  TRID = delaunay(X,Y,lt)
```

Voici la triangulation résultante pour le schéma d'échantillonnage proposé et pour l'exemple considéré:



Et voici une amplification permettant d'observer les structures triangulaires plus en détail :



Étape 3: *Calcul des gradients pour chaque sommet*

Une fois que l'on a calculé la triangulation en 2D, on transforme nos triangles en 2D en des triangles en 3D en considérant non plus les sommets de chaque triangle comme des points de coordonnées (x,y) mais comme des points de coordonnées (x,y,z), z étant la valeur des pixels en chaque point (x,y). Ceci nous donne un ensemble de triangles en trois dimensions sur lesquels on va calculer le gradient en chacun des sommets.

On utilise la formule proposée dans l'article, selon laquelle le vecteur «normal» en chaque sommet est la moyenne des vecteurs normaux des triangles adjacents à ce sommet pondérés par leurs surfaces :

$$\vec{v}_s = \sum_{j=1}^{x} \frac{A_j \vec{n}_j}{A} \qquad A = \sum_{j=1}^{x} A_j$$

où Aj est la surface du j-ième triangle voisin du sommet «s» et x est le nombre de triangles qui partagent le sommet « s »

Ce qui nous permet d'obtenir les gradients selon les directions x et y à partir des expressions :

$$\frac{\partial z}{\partial x} = -\frac{n_x}{n_z} \qquad \frac{\partial z}{\partial y} = -\frac{n_y}{n_z}$$

Pour le calcul des vecteurs normaux et des surfaces de chaque triangle adjacent à un sommet, nous avons créé notre propre fonction Matlab, «*area3d*», qui utilise la méthode classique du produit vectoriel.

Les autres calculs, ceux du vecteur normal et du gradient en chaque sommet, ont été codés dans la fonction «*get_gradients*», qui utilise la fonction «*area3d*».

***Étape 4:* Calcul des coefficients des surfaces d'approximation**

Ayant calculé les vecteurs normaux pour tous les sommets intervenant dans la triangulation 3D de notre image, on va essayer de modéliser de façon approximative les divers triangles par des surfaces «lisses».

On utilisera comme outil des polynômes d'ordre trois avec deux variables x et y:

$$z(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 y^2 + c_6 x^3 + c_7 x^2 y + c_8 x y^2 + c_9 y^3$$

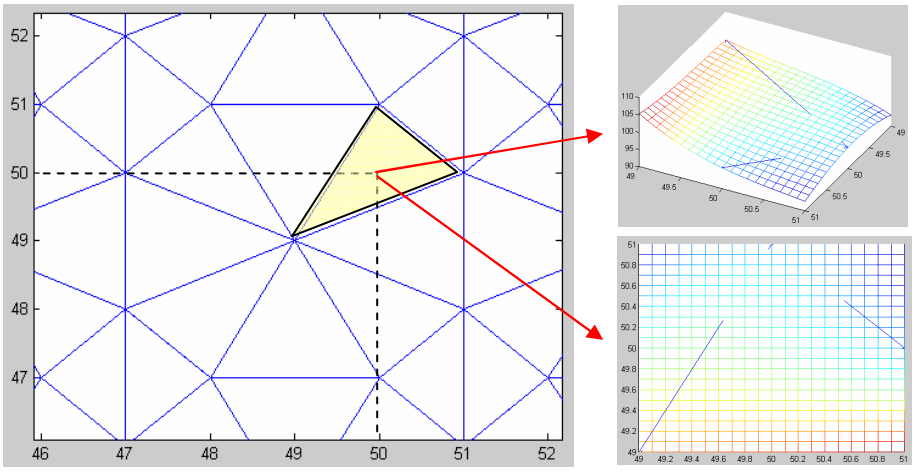
Cette expression a neuf coefficients inconnus que l'on pourra déterminer à partir des valeurs de niveau de gris de l'image sur les sommets de chaque triangle et à partir des valeurs des gradients que l'on a calculés dans la section précédente. On obtiendra ainsi un système d'équations pour chaque région triangulaire du plan originel de l'image sur lequel on avait appliqué l'algorithme de Delaunay.

Pour chaque sommet d'un triangle on peut écrire un système d'équations de la forme:

$$\begin{bmatrix} z_i \\ \frac{\partial z_i}{\partial x} \\ \frac{\partial z_i}{\partial y} \end{bmatrix} = \begin{bmatrix} 1 & x_i & y_i & x_i^2 & y_i^2 & x_i^3 & x_i^3 & x_i y_i^2 & y_i^3 \\ 0 & 1 & 0 & 2x_i & 0 & 3x_i^2 & 3x_i^2 & y_i^2 & 0 \\ 0 & 0 & 1 & 0 & 2y_i & 0 & 0 & 2x_i y_i & 3y_i^2 \end{bmatrix} \cdot c$$

et donc, les trois sommets associés à chaque triangle nous permettront d'obtenir les neufs coefficients cherchés.

Exemple pour le triangle contenant le point (50,50) :



La résolution d'un tel système en Matlab est effectuée grâce à la commande 'pinv', basée sur le calcul de la matrice inverse de Moore-Penrose et capable d'obtenir des solutions pour les matrices singulières avec une erreur quadratique minimale en norme.

***Étape 5:* Calcul des nouvelles valeurs de niveau de gris et reconstruction de l'image**

La dernière étape pour l'implémentation de l'algorithme de superresolution est celle consistant à calculer les valeurs de niveau de gris en chacun des points de l'image finale à l'aide des polynômes dont les coefficients ont été obtenus à la fin de l'étape précédente.

Cette étape, plutôt triviale du point de vue conceptuel, est intéressante du point de vue de l'implémentation en Matlab.

En effet, cette étape nécessite de retrouver le triangle contenant un point donné parmi tous les triangles formés par la triangulation Delaunay. Bien que la commande «*tsearch*» ait cette fonction, on a constaté qu'elle présentait quelques problèmes pour la localisation de certains points. En effet, même si un point est contenu dans un triangle, la commande «*tsearch*» retourne parfois NaN (*not-a-number*) (versions 6 et 7 de Matlab).

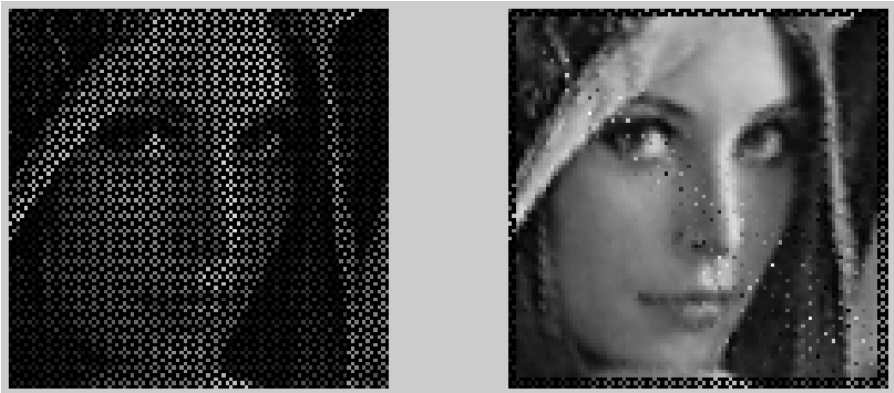
On a donc codé cette recherche «manuellement», à l'aide de la commande «*dsearch*» qui a pour fonction de trouver le sommet d'un triangle le plus proche du point donné. Ensuite, on teste l'appartenance du point cible à un triangle, pour chacun des triangles qui possèdent le sommet trouvé, à l'aide de la commande «*inpolygon*». On peut rencontrer des situations dans lesquelles le point le plus proche du point cible soit le sommet d'un triangle adjacent à celui qui contient le point cible en question : la boucle '*while*' s'occupe de parcourir tous les triangles voisins afin de trouver le bon.

Voici le code Matlab développé:

```
T=tsearch(X,Y,TRID,px,py);


% tsearch "manuel"
if (isnan(T))
   T=[];
   k=dsearch(X,Y,TRID,px,py);   % on cherche le vertex le plus proche
% on cherche tous les triangles pour ce vertex

  index = find( TRID(:,1)==k | TRID(:,2)==k | TRID(:,3)==k);
  for i=1:length(index)
      % on determine celui ou ceux qui contiennent le point
      in = inpolygon(px,py,X(TRID(index(i),:)),Y(TRID(index(i),:)));
            if (in==1)
         T= [ T index(i)];
      end
   end

   if (numel(T)==0)

      X_AUX=X;
      Y_AUX=Y;
```

```
      while (numel(T)==0)
         X_AUX=[X_AUX(1:k-1) -100 X_AUX(k+1:length(X_AUX))];
         Y_AUX=[Y_AUX(1:k-1) -100 Y_AUX(k+1:length(Y_AUX))];

         k=dsearch(X_AUX,Y_AUX,TRID,px,py);

         % on cherche tous les triangles pour ce vertex
         index = find( TRID(:,1)==k | TRID(:,2)==k | TRID(:,3)==k);
         for i=1:length(index)
            in = inpolygon(px,py,X(TRID(index(i),:)),Y(TRID(index(i),:)));
         % on détermine celui ou ceux qui contiennent le point
            if (in==1)
               T= [ T index(i)];
            end
         end

      end
   end

end
```

Ci-dessous, le résultat obtenu pour la reconstruction de l'image de Lena (100x100 pixels) à partir des 5 imagettes (25x25 pixels) :



Note: Dans la fonction « *reconstruct* », qui s'occupe de calculer les valeurs de l'image pour les nouveaux points ajoutés à partir des surfaces approximées obtenues, on ignore les trois premières et les trois dernières positions horizontales et verticales, afin d'éviter des particularités sur les bords de la triangulation (si on commence au point [2,2], par exemple, avec un pas d'échantillonnage de 4 pixels, on arrivera jusqu'au point [2,98], et si l'on tente de chercher le point [2,99] dans la triangulation de base on ne le trouvera pas). C'est pourquoi les images ici reconstruites ont un « cadre » qui les entoure. On a laissé de coté ce problème qui nous a semblé être un détail de programmation face aux concepts mathématiques décrits dans la procédure.

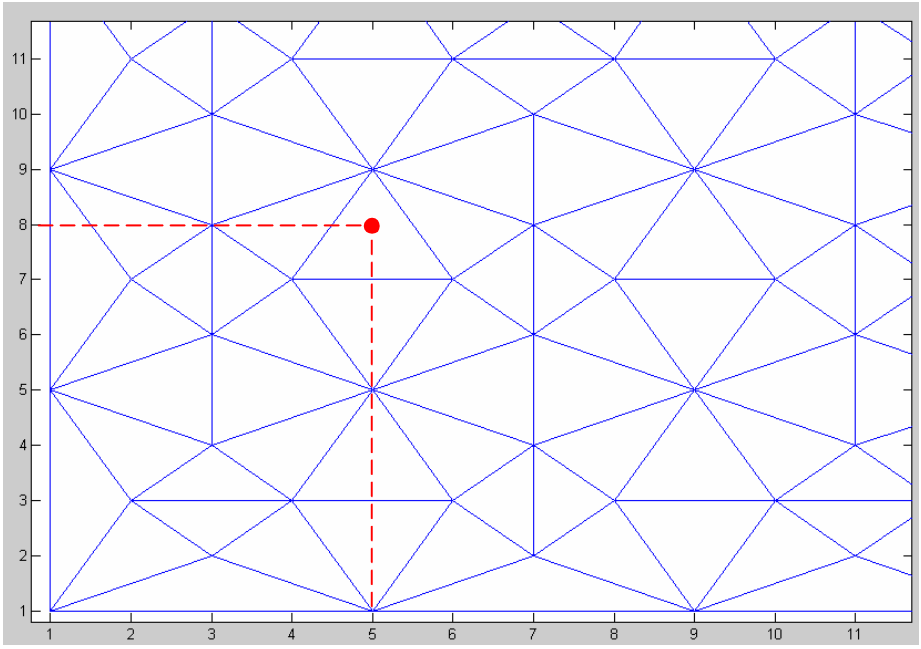
L'algorithme d'insertion

Lerttratanapanich et Bose proposent dans leur article un algorithme très intéressant pour l'insertion de nouvelles imagettes de basse résolution permettant d'améliorer la qualité d'une image de «haute» résolution construite par la méthode que l'on vient de décrire.

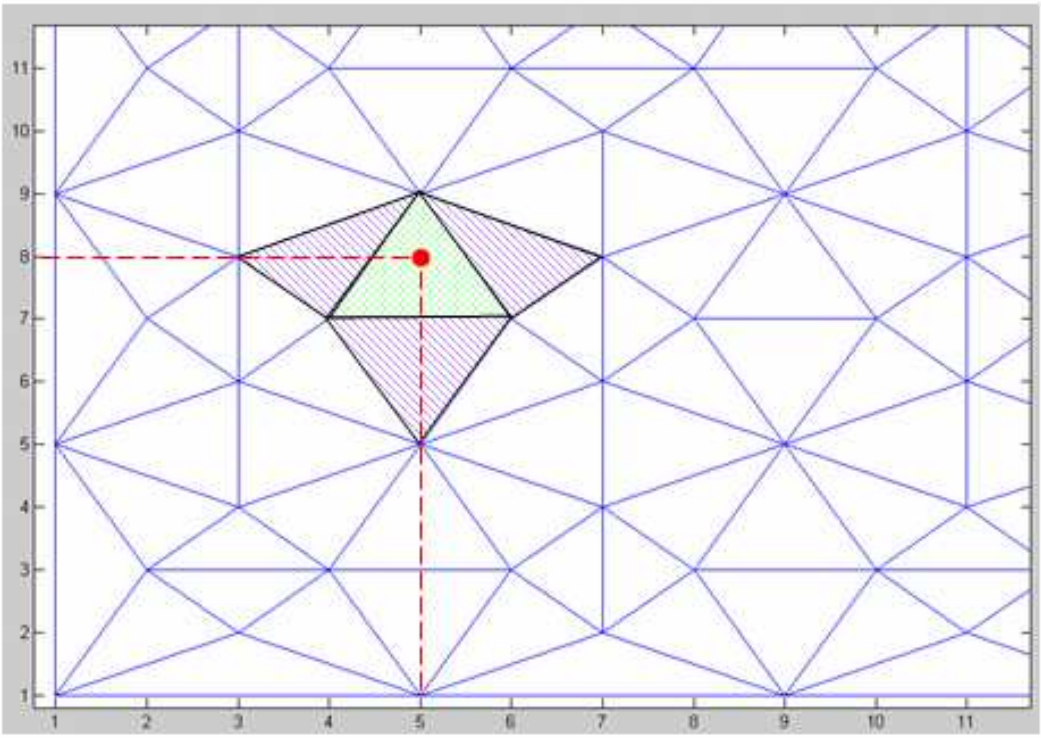
Cet algorithme utilise des propriétés mathématiques qui évitent de reconstruire la triangulation de Delaunay en entier, ce qui représente un gain important en terme de complexité. En effet, l'ajout d'une imagette de basse résolution entraînera l'insertion de nouveaux points dans le schéma d'échantillonnage existant. Ce qui implique, que des triangles ne seront plus Delaunay car leurs cercles circonscrits contiendront ces nouveaux points. D'autres triangles, par contre, ne seront pas affectés et il ne sera donc pas nécessaire ni de les modifier, ni de recalculer les gradients et les surfaces d'approximation associés.

On a implémenté cet algorithme, que l'on va expliquer ci-dessous avec un exemple précis, en Matlab dans les fonctions «*add_LR*» et «*add_point*» (code téléchargeable sur la page d'accueil).

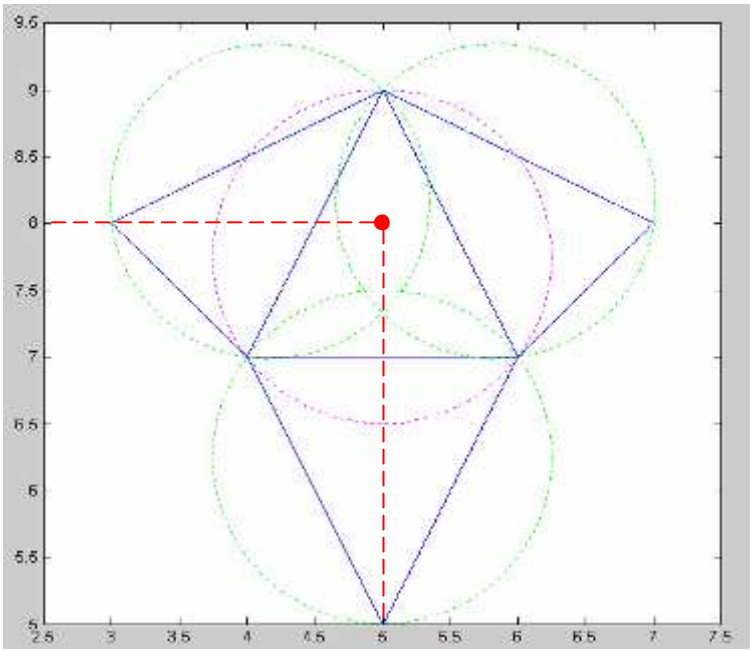
Considérons d'abord la triangulation décrite précédemment. On ajoute le point d'échantillonnage (5,8) qui ne correspond à aucun des sommets existants dans la triangulation Delaunay que l'on avait calculée :



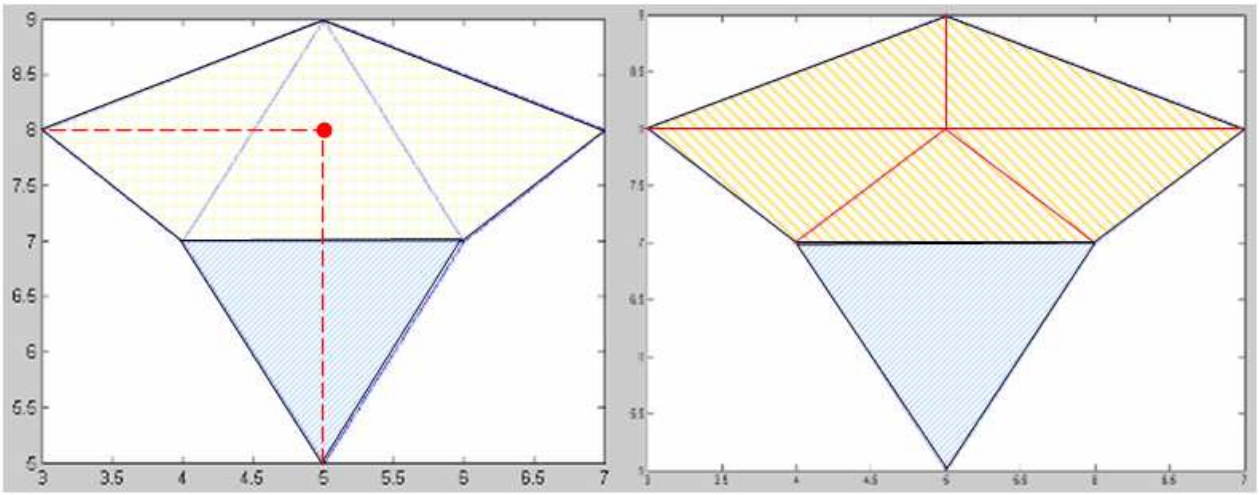
Premièrement, on cherche le triangle qui contient le point en question. Puis, on considère tous les triangles adjacents à celui-ci, c'est à dire, tous ceux partageant une arête avec lui :



On obtient ainsi quatre triangles pour lesquels on va vérifier la propriété de Delaunay.



On considère les trois triangles, dans ce cas présent, qui ne sont plus Delaunay (dont le cercle circonscrit contient le nouveau point, en jaune sur l'image) :

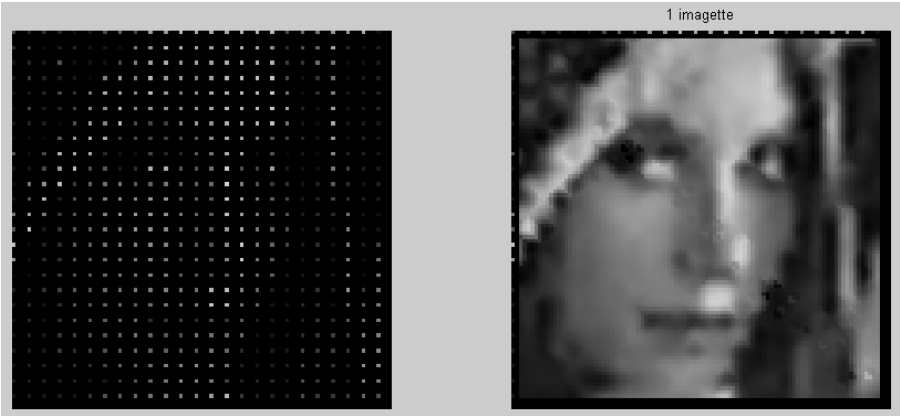


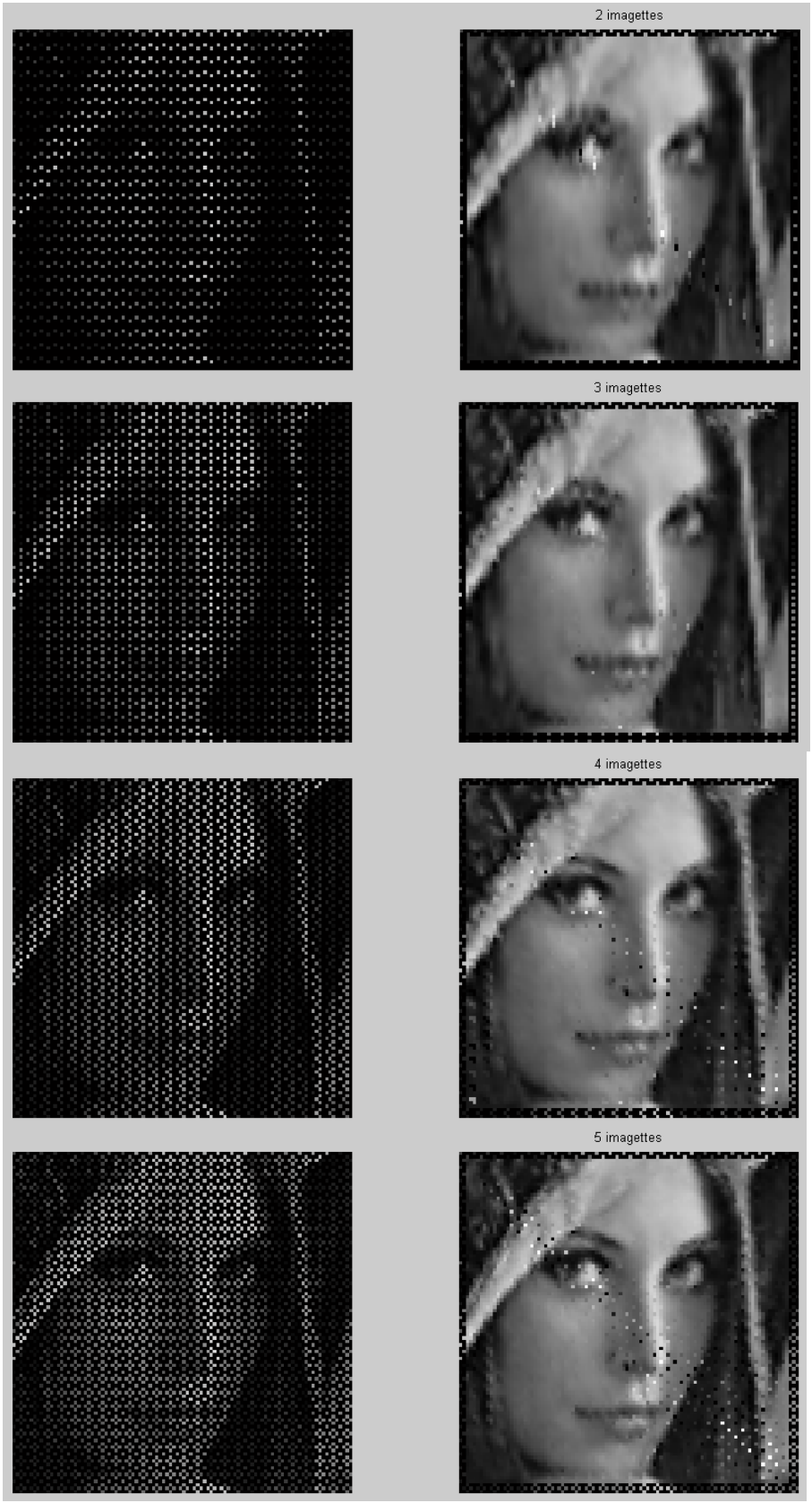
On réalise une nouvelle triangulation Delaunay reliant les sommets de ces trois triangles au nouveau point ajouté. C'est ainsi que l'ajout de ce nouveau point au schéma d'échantillonnage, nous amène à remplacer trois des anciens triangles par cinq nouveaux (en orange).

On devra alors recalculer les vecteurs gradients associés à ces nouveaux triangles, ainsi que leurs surfaces d'approximation.

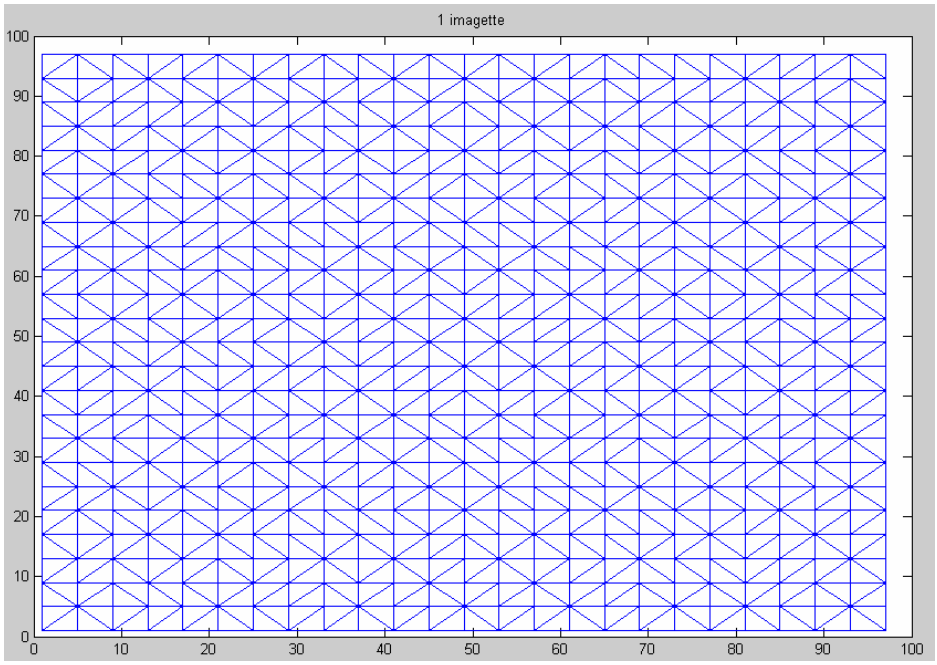
Cet algorithme sera répété pour tous les points de la nouvelle image de basse résolution que l'on souhaitera ajouter.

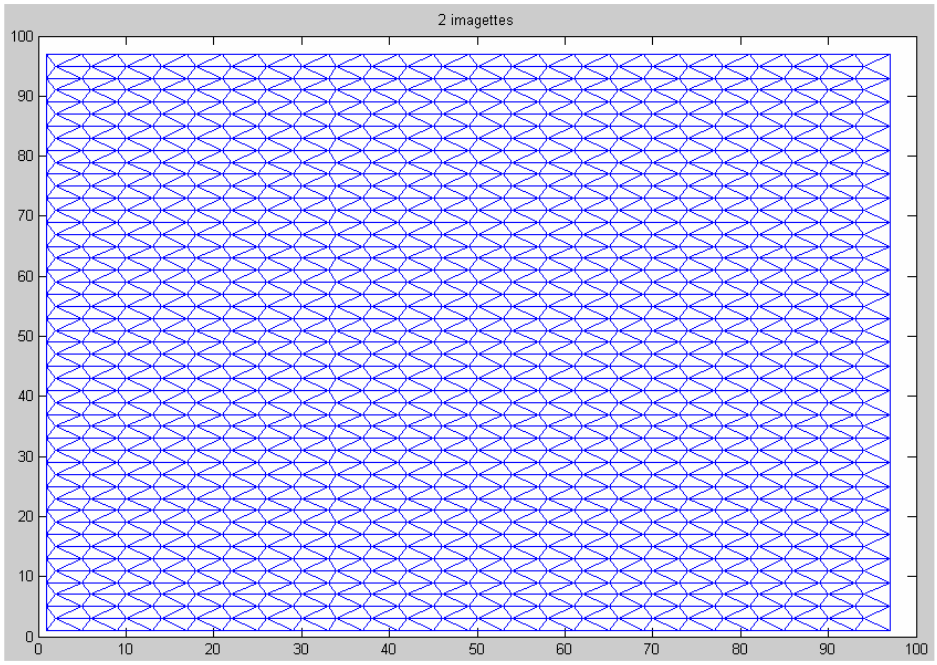
On obtient ainsi une image de haute résolution qui s'améliore avec le nombre d'imagettes de basse résolution insérées au processus, comme on peut le constater avec les résultats ci-dessous :

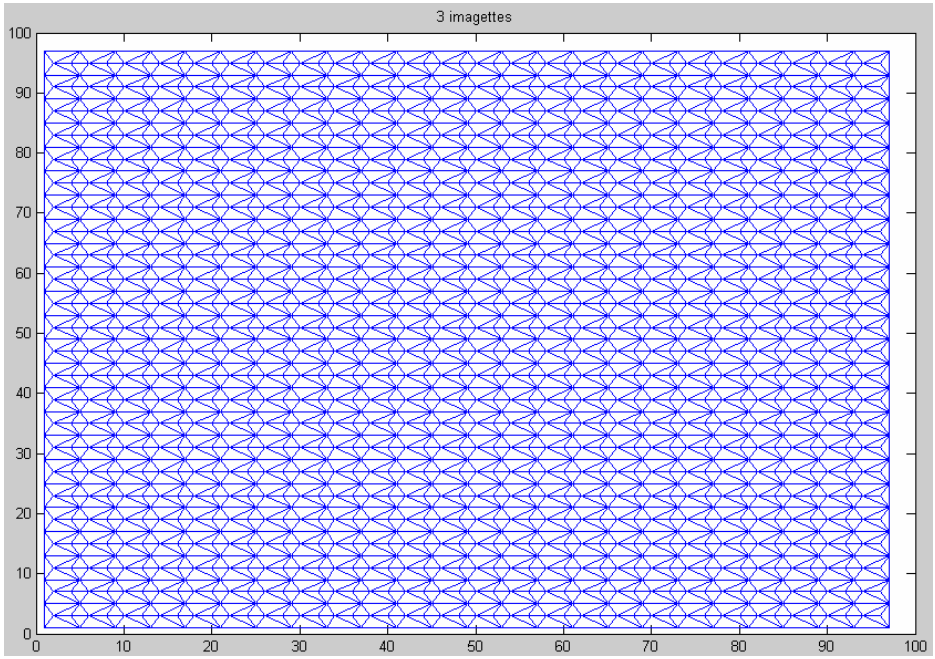


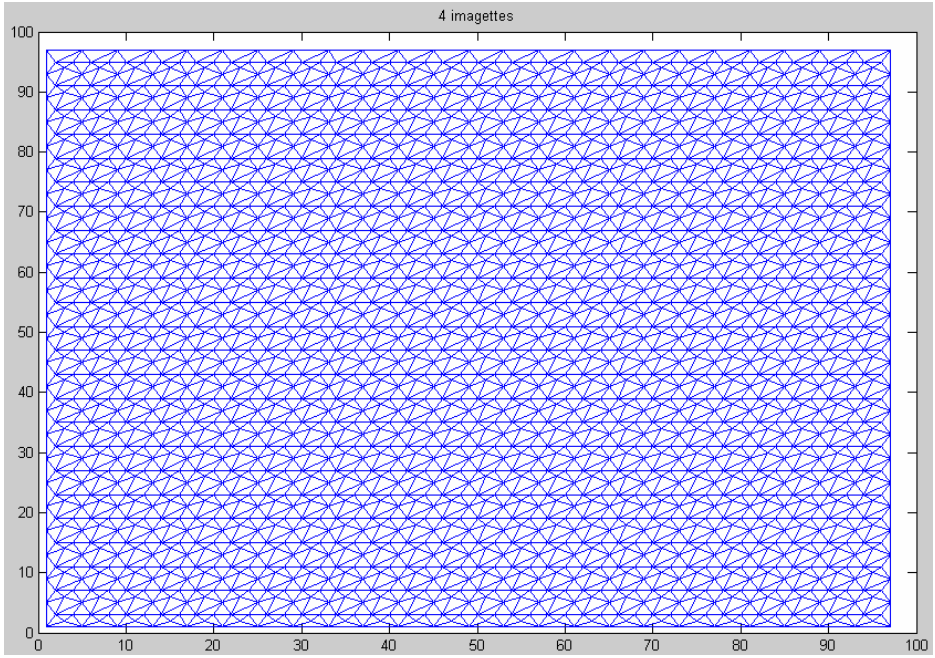

2 imagettes
3 imagettes
4 imagettes
5 imagettes

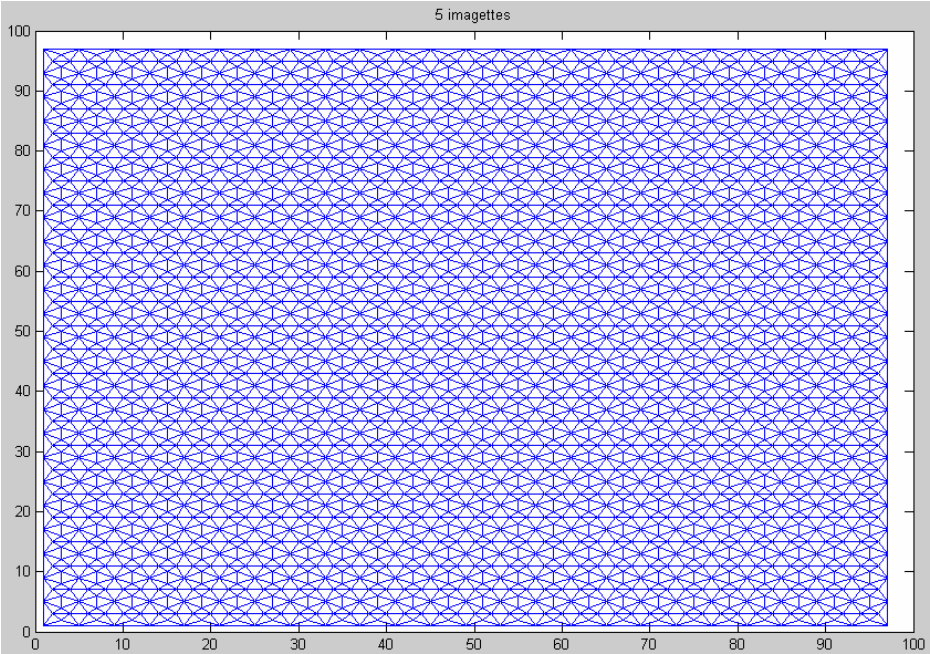
Et voici l'évolution des triangulations sous-jacentes :



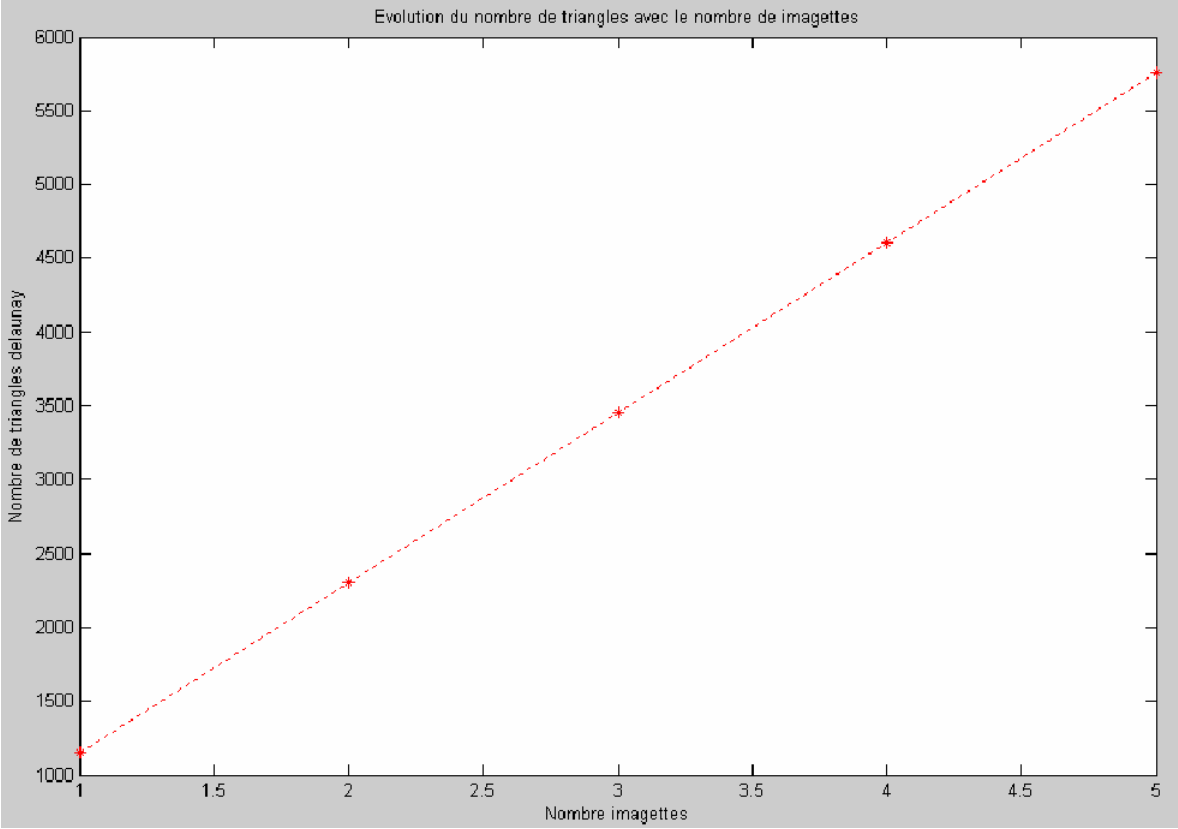



3 imagettes
0
10
20
30
40
50
60
70
80
90
100


4 imagettes
0
10
20
30
40
50
60
70
80
90
100

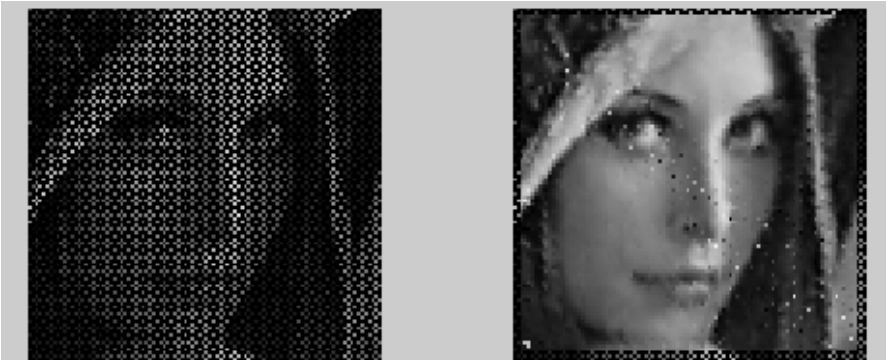


Comme on s'y attendait, on observe une évolution linéaire du nombre de triangles avec le nombre d'imagettes:



Le bruit de reconstruction

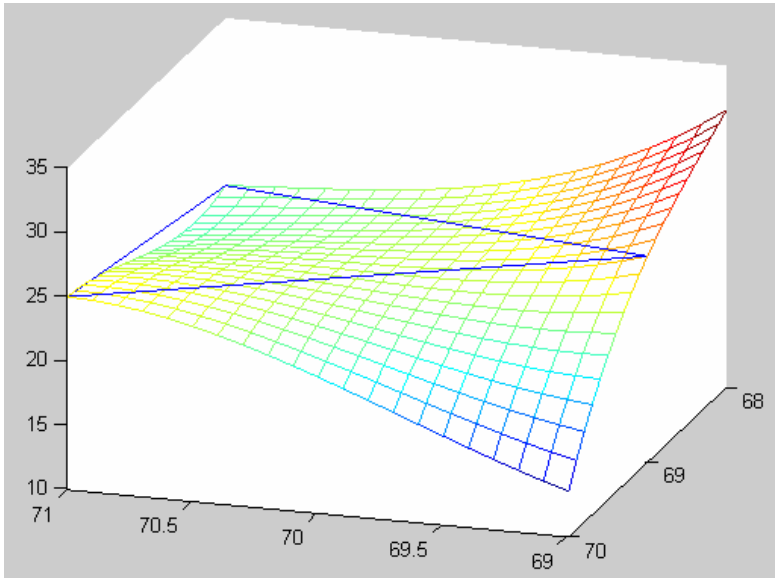
Dans les résultats précédents, on peut observer l'apparition d'un certain nombre de points bruités qui se concentrent clairement sur la diagonale principale de l'image de haute résolution construite :



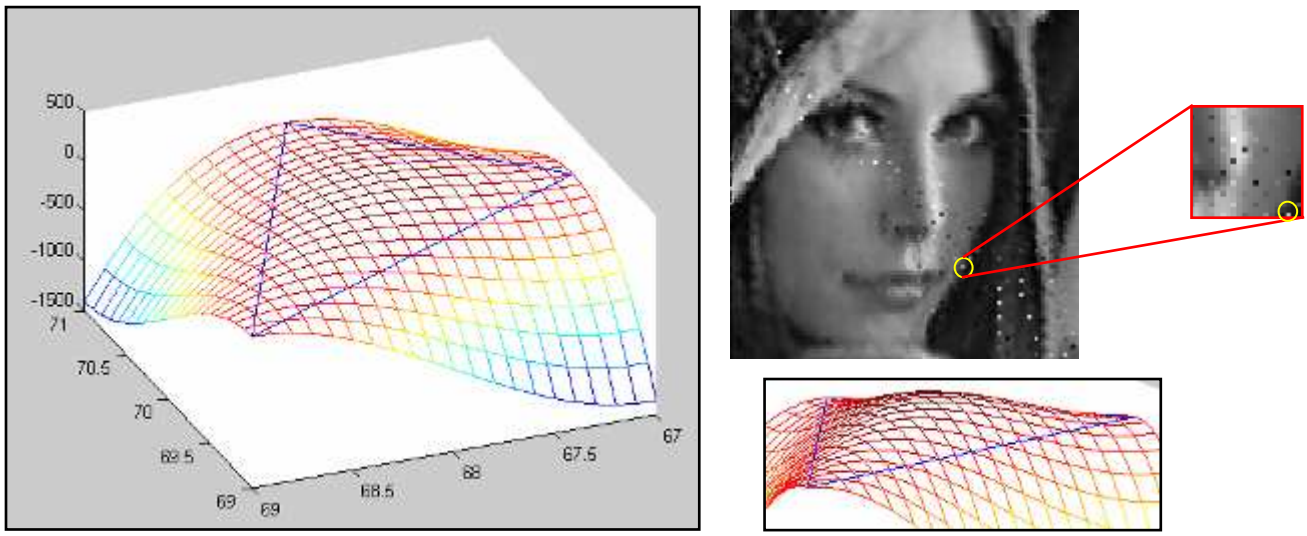
On va essayer de comprendre pourquoi ce bruit existe et pourquoi il apparaît sur la diagonale principale en regardant l'allure des surfaces d'approximation obtenues pour deux points voisins, l'un étant bien reconstruit et l'autre bruité.

On va prendre ces points dans une région plutôt obscure entre le visage de Lena et le fond (voir figure ci-dessous). On s'attend donc à obtenir des valeurs d'intensité plutôt basses, c'est-à-dire des valeurs numériques petites pour les niveaux de gris.

Si l'on regarde la surface d'approximation pour le point bien reconstruit, on observe que cette surface s'adapte bien à l'orientation spatiale du triangle de référence :



Si, par contre, on regarde le cas du point voisin bruité, on obtient une situation très différente :



Dans ce cas-là, l'approximation surfacique colle mal au triangle considéré. On observe l'apparition d'une bosse, une «petite montagne», qui provoque l'apparition d'un point très claire dans une région plutôt sombre dans la reconstruction de l'image de haute résolution. Pour des valeurs localisées près du centre géométrique du triangle, l'approximation surfacique est très mauvaise. Par contre les contours, les valeurs de gris et les gradients calculés pour les trois sommets sont bien respectés.

Lettratanapanich et Bose mentionnent cet aspect comme un des inconvénients majeurs de l'algorithme, et il semble qu'il provienne de l'utilisation du produit vectoriel pour l'estimation du gradient à chaque sommet. En effet, cette estimation dépend fortement de l'ensemble des triangles qui partagent le sommet et par conséquent la qualité de l'approximation surfacique, obtenue à l'aide d'un polynôme bicubique, en dépend aussi. Les contraintes prises en compte concernent seulement les sommets de la région triangulaire, et ignorent, par exemple, les arêtes, ce qui peut être à l'origine d'une surface pas suffisamment «lisse», comme celle que l'on a d'observer plus haut.

Quand on utilise une ou deux imagettes seulement, ce phénomène est moins évident, puisqu'il apparaît seulement pour certaines orientations spatiales des triangles à approximer. Plus le nombre de ces triangles est grand, plus la probabilité de trouver des points bruités est élevée.

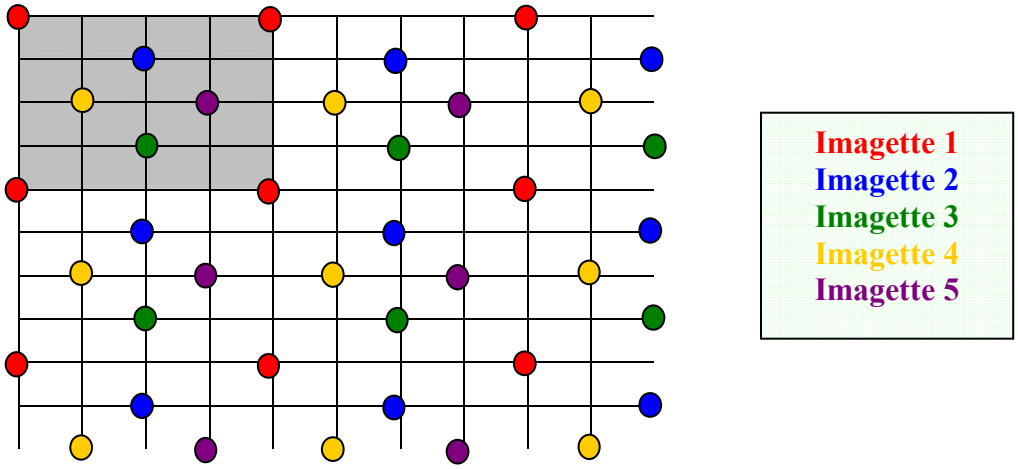
Ceci dit, on a observé une influence très importante du choix pour le schéma d'échantillonnage sur la qualité de la reconstruction et l'apparition des points bruités. En effet, la répartition géométrique des points d'échantillonnage peut modifier de façon conséquente le résultat de l'image de haute résolution, au point, soit d'éliminer presque tous les points bruités, soit de détériorer fortement l'image (il s'agit alors de schémas que l'on pourrait considérer «pathologiques»).

On va illustrer ce point avec trois exemples de schéma d'échantillonnage différents pour l'image de Lena.

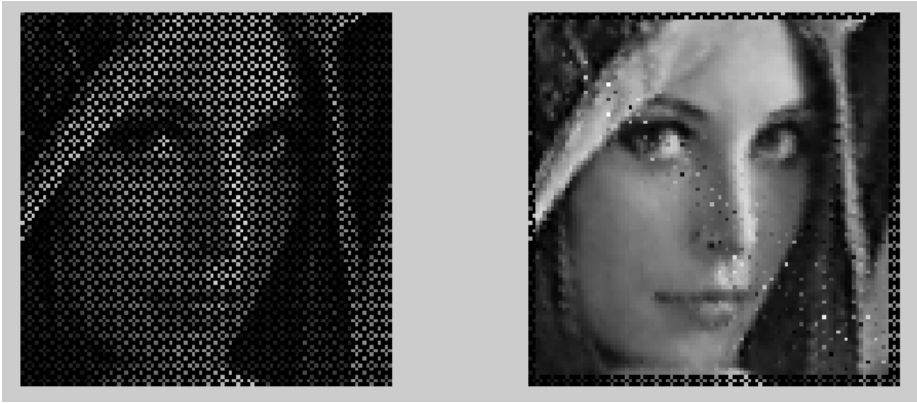
Exemple A

On reprend, tout d'abord, l'exemple considéré jusqu'à présent :

Schéma d'échantillonnage :



Résultat après reconstruction :

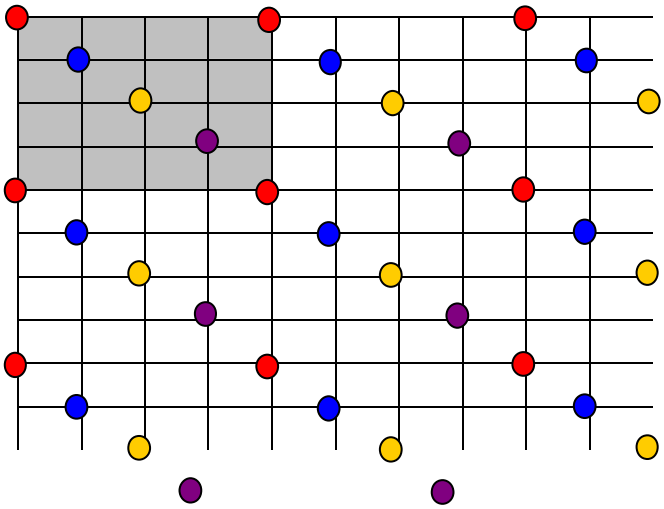


On observe à nouveau le bruitage près de la diagonale de l'image, comme on l'a déjà expliqué.

Exemple B

On choisit de prendre nos points d'échantillonnage suivant une diagonale, pour voir ce qu'il se passe…

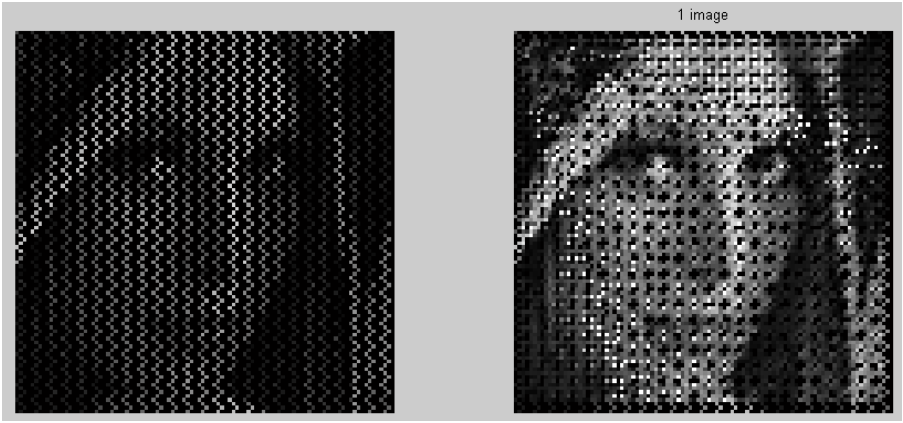
Nouveau schéma d'échantillonnage (pour 4 imagettes):



Imagette 1
Imagette 2
Imagette 3
Imagette 4

Résultat après reconstruction :

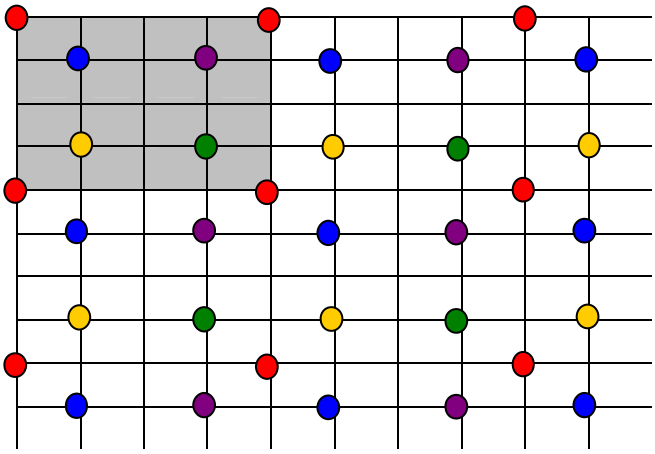
On observe que les conséquences sont désastreuses.



Exemple C

Pour finir, on choisit un schéma d'échantillonnage bien distribué qui recouvre l'espace de l'image originelle.

Nouveau schéma d'échantillonnage :



Imagette 1
Imagette 2
Imagette 3
Imagette 4
Imagette 5

Résultat :

Les points bruités ont significativement disparus et l'image gagne par conséquent en qualité.

Conclusion :

Avec une quantité d'information similaire (4 ou 5 imagettes de 25x25 pixels), la différence entre les résultats observés pour les exemples A, B et C est flagrante.

Il est donc, tout à fait, raisonnable d'affirmer que le choix du schéma d'échantillonnage a une influence décisive sur la qualité de l'image reconstruite.

Résultats des tests sur des différents types d'images

Dans cette section, on utilisera le schéma d'échantillonnage de l'exemple C de la section antérieure, puisqu'il offre un bon compromis entre la qualité de la reconstruction et la quantité d'imagettes intervenant dans le processus.

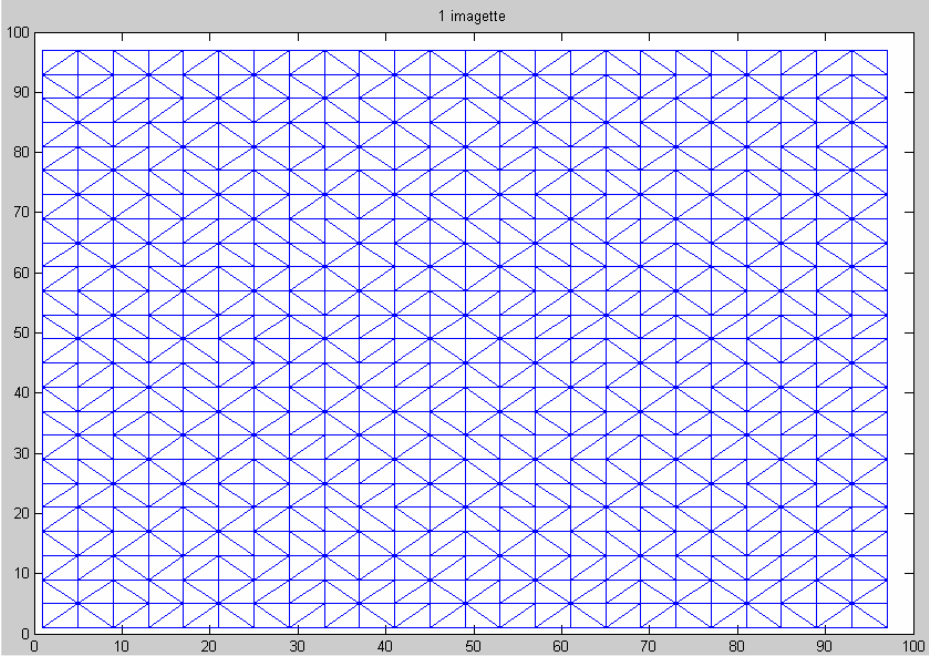
On va considérer quatre types d'images pour nos tests :

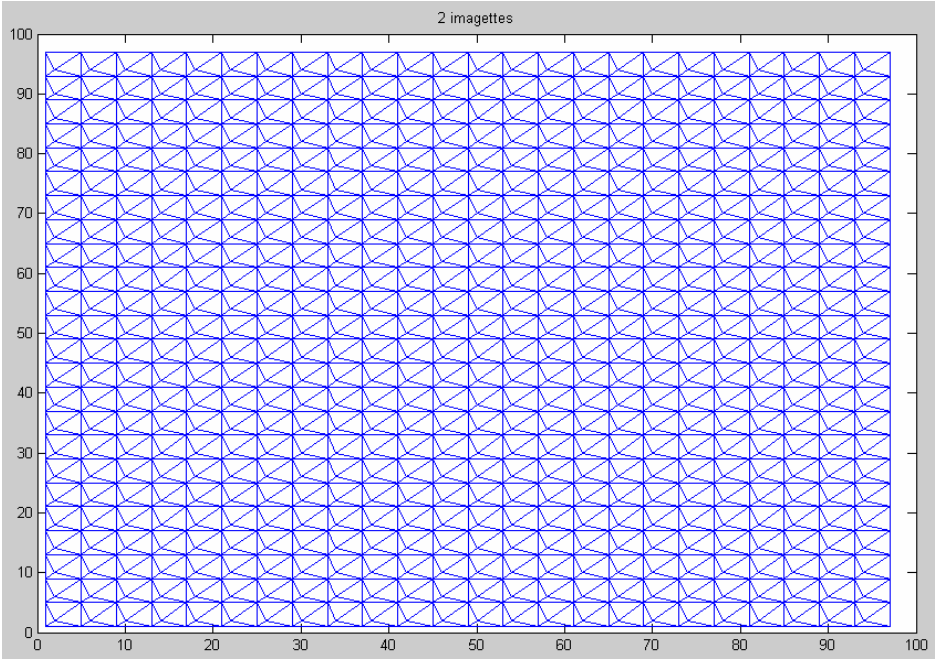
Une image classique : Lena
Une image médicale : Cerveau
Une image satellite : SPOT
Une photo prise avec une camera numérique : Entrée de l'ENST

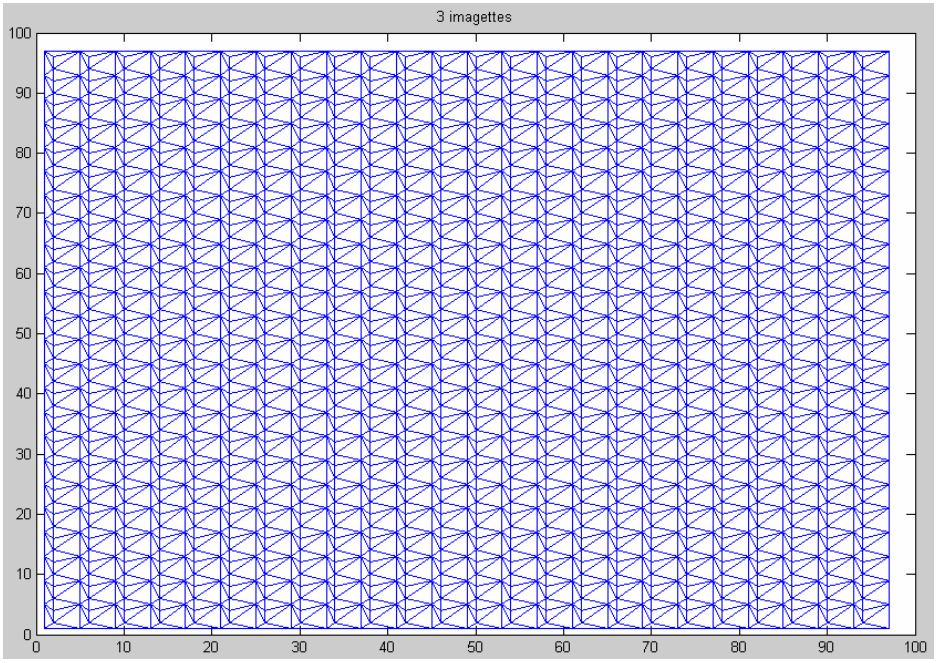
Pour chacune de ces images, on appliquera l'algorithme de superresolution. Puis, on montrera les résultats des insertions successives, cinq au total, permettant d'arriver au résultat «final». Ensuite, on regardera l'évolution de l'erreur quadratique moyenne MSE avec et sans filtrage médian (celui implémenté dans le toolkit de traitement d'image de Matlab), ainsi que la variation des resultats avec les differents valeurs du parametre associée à celui-ci.

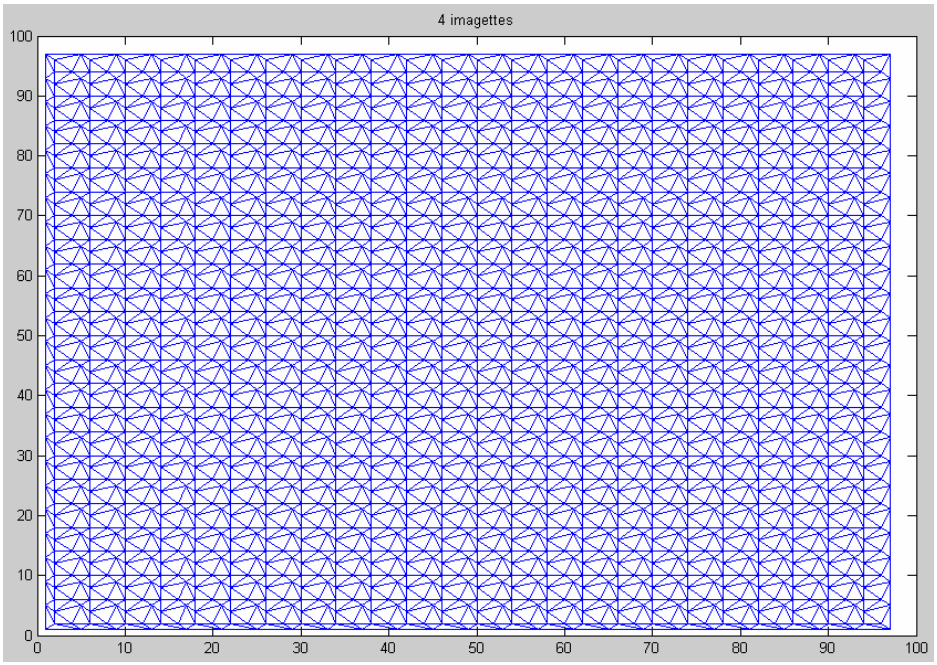
Evolution des triangulations

On commencera par montrer l'évolution des triangulations delaunay 2D pour le schéma choisi et pour une région de 100x100 pixels:




2 imagettes
0
10
20
30
40
50
60
70
80
90
100


3 imagettes
0
10
20
30
40
50
60
70
80
90
100



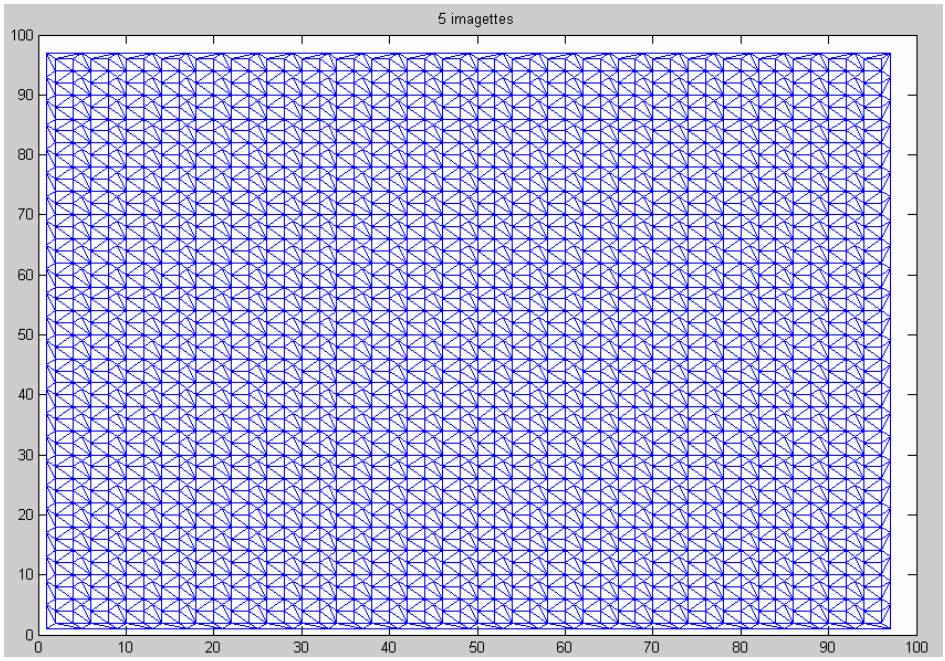
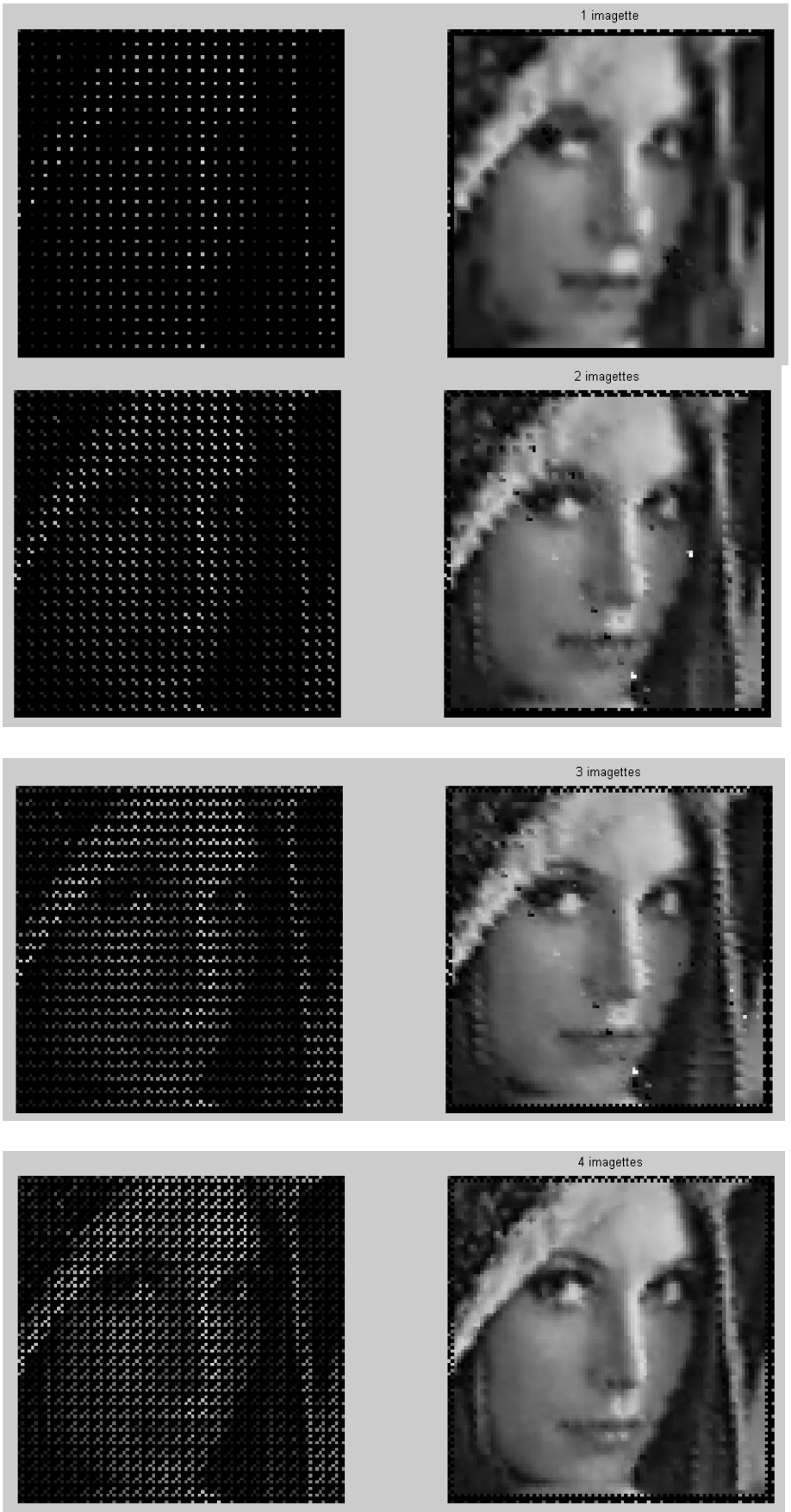
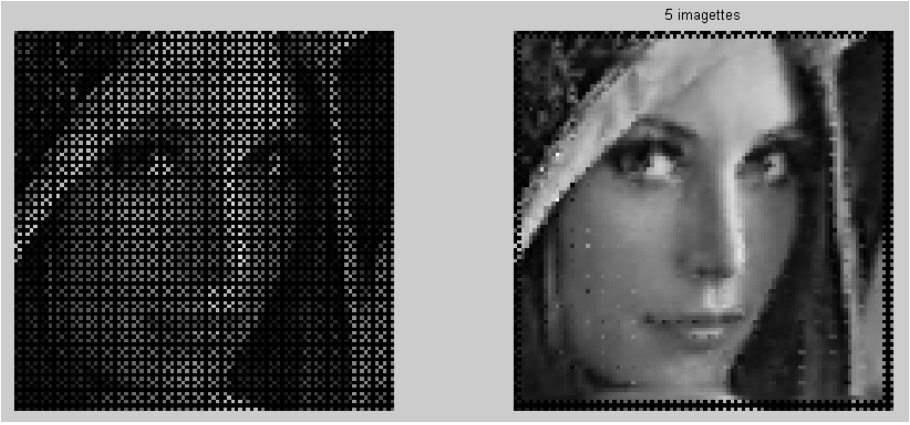


Image 1: Lena

Voici les resultats pour la reconstruction progressive d'un morceau central de taille 100x100 pixels de l'image classique Lena avec le schéma d'échantillonnage choisi:


1 imagette
2 imagettes
3 imagettes
4 imagettes



Et voici les résultats avec et sans filtrage médian (on a testé plusieurs filtres médian) de l'image reconstruite au fur et à mesure de l'ajout d'imagettes dans la reconstruction:

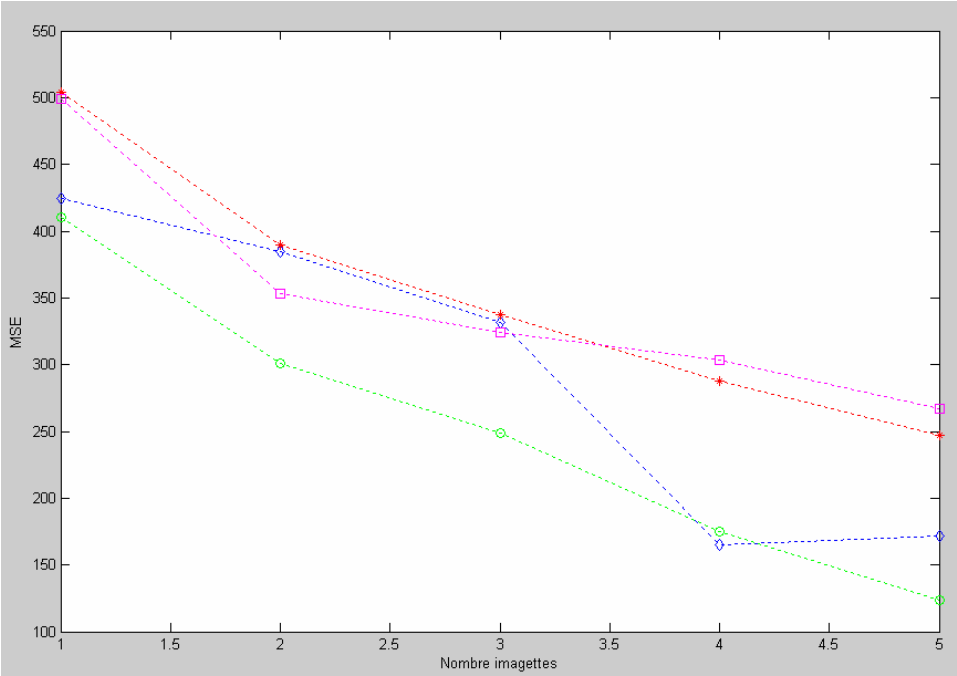
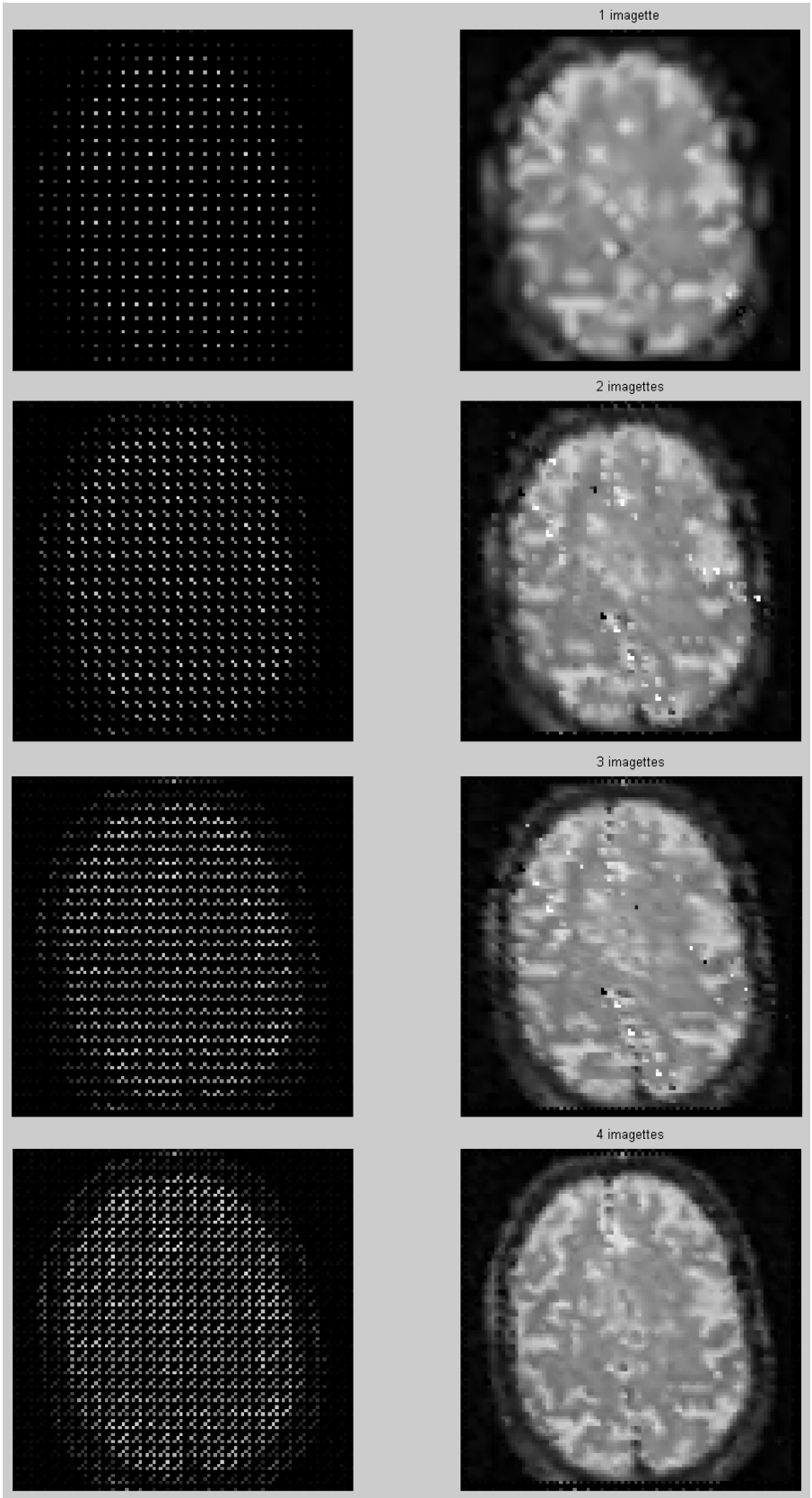
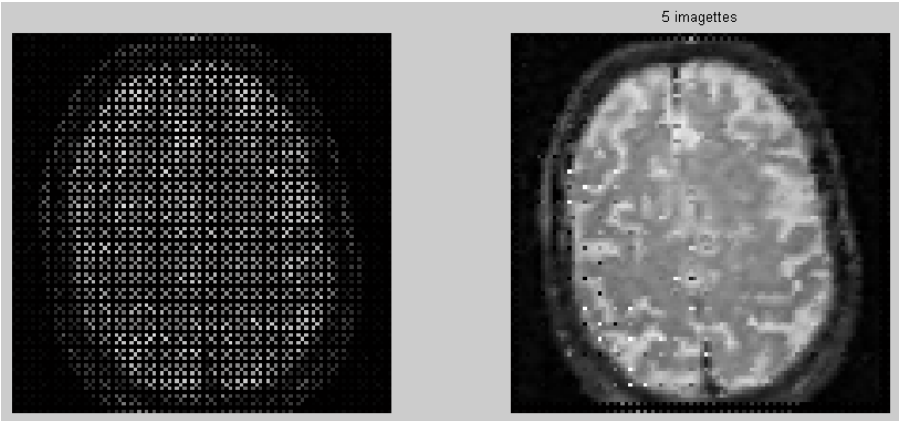


Image 2: imagerie médicale cerveau

On travaille maintenant avec une image 100x100 pixels du cerveau.

Voici l'image « haute résolution » originale et les reconstructions successives:


1 imagette
2 imagettes
3 imagettes
4 imagettes



Et voici les résultats avec et sans filtrage médian de l'image reconstruite au fur et à mesure de l'ajout d'imagettes dans la reconstruction:

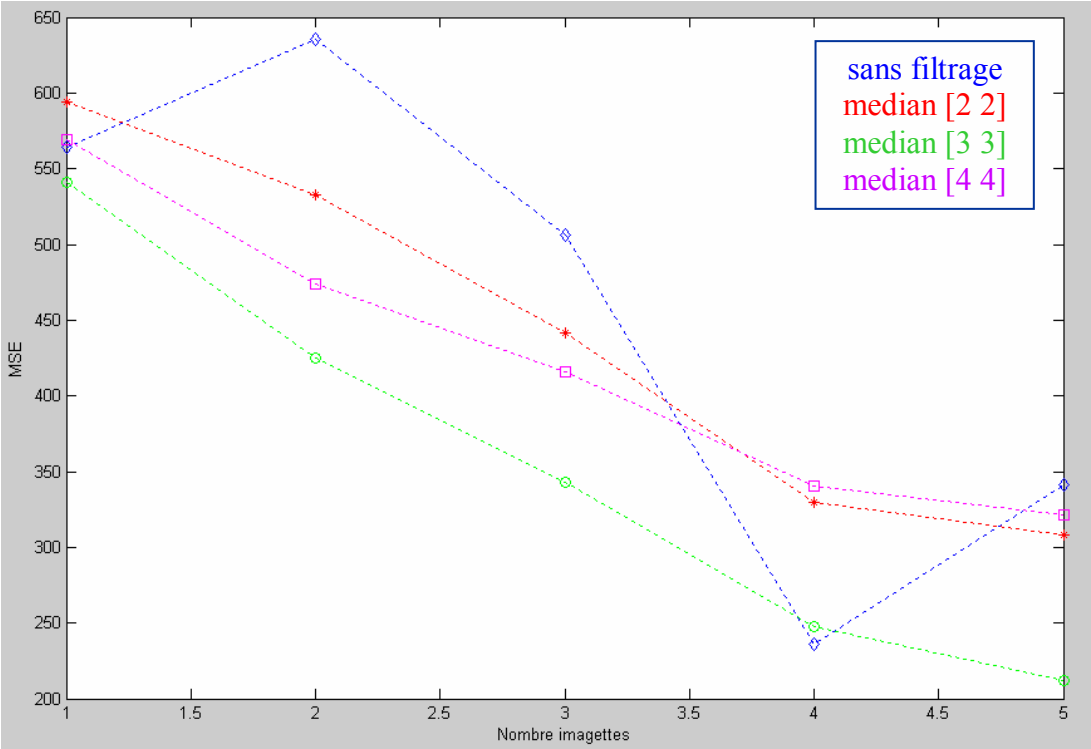


Image 3: imagerie satellite SPOT

Notre troisième image provient des TP du module ADES de la brique Reconnaissance de Formes (RdF) à l'ENST pour le T1 de l'année académique 2003/2004.

Il s'agit d'une image du satellite SPOT sur laquelle on a pris une région de 256x256 pixels pour nos simulations.

Voici l'image « haute résolution » originale et les reconstructions successives:


2 imagettes
50
100
150
200
250
50
100
150
200
250


3 imagettes
50
100
150
200
250
50
100
150
200
250


4 imagettes
50
100
150
200
250
50
100
150
200
250


5 imagettes
50
100
150
200
250
50
100
150
200
250

Et voici les résultats avec et sans filtrage médian de l'image reconstruite au fur et à mesure de l'ajout d'imagettes dans la reconstruction:

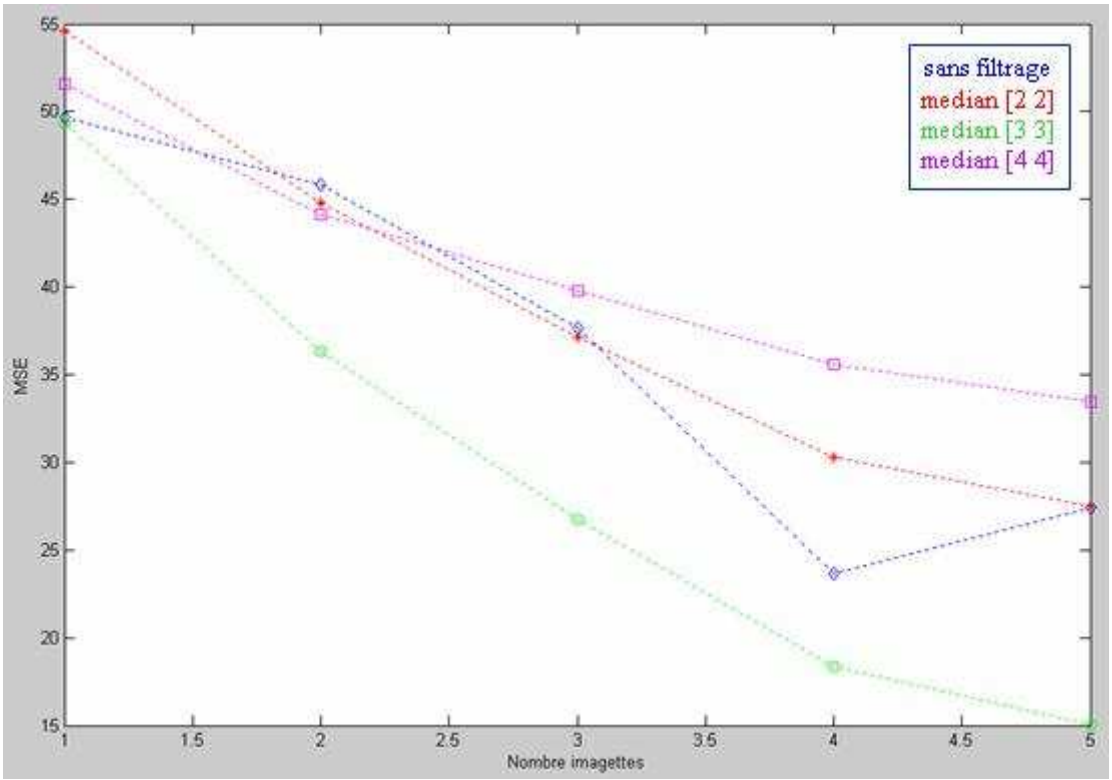
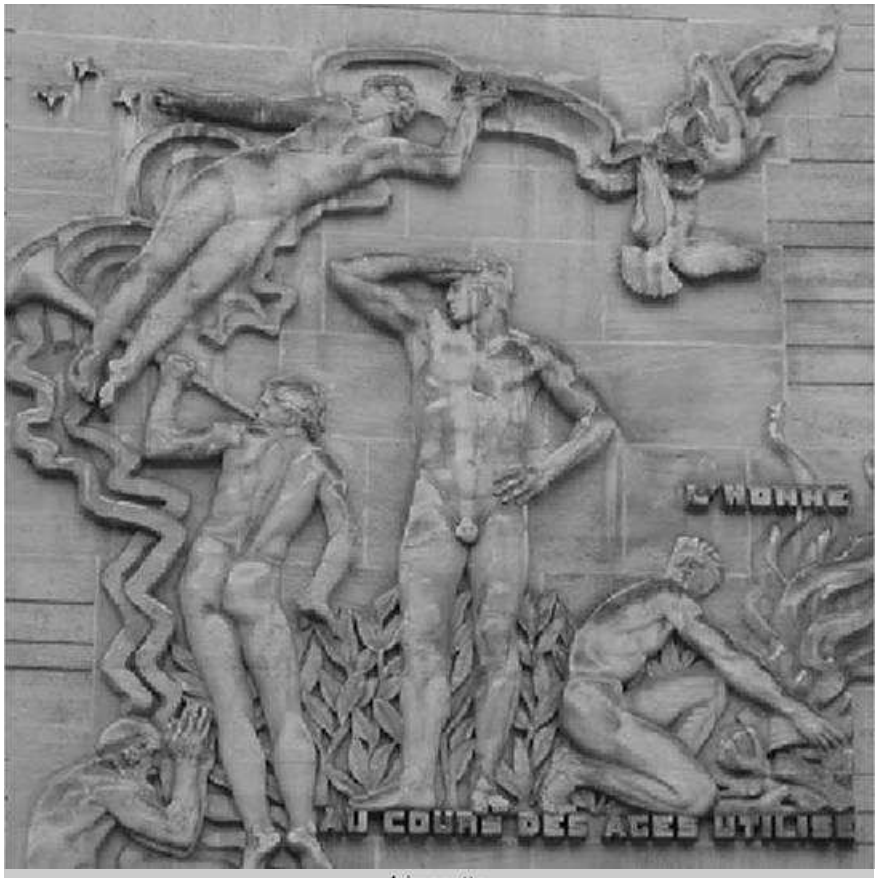


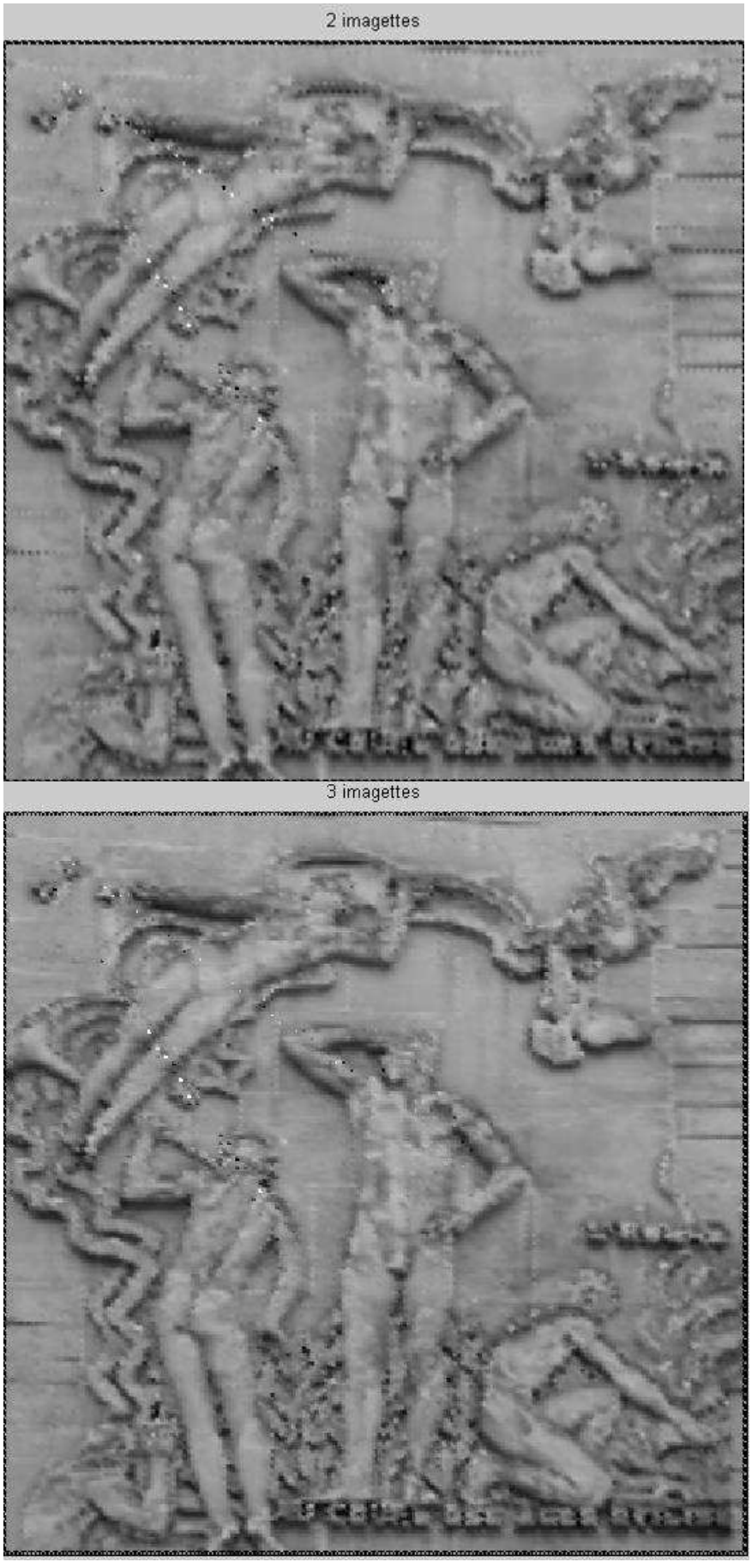
Image 4: photo numérique de l'entrée de l'ENST

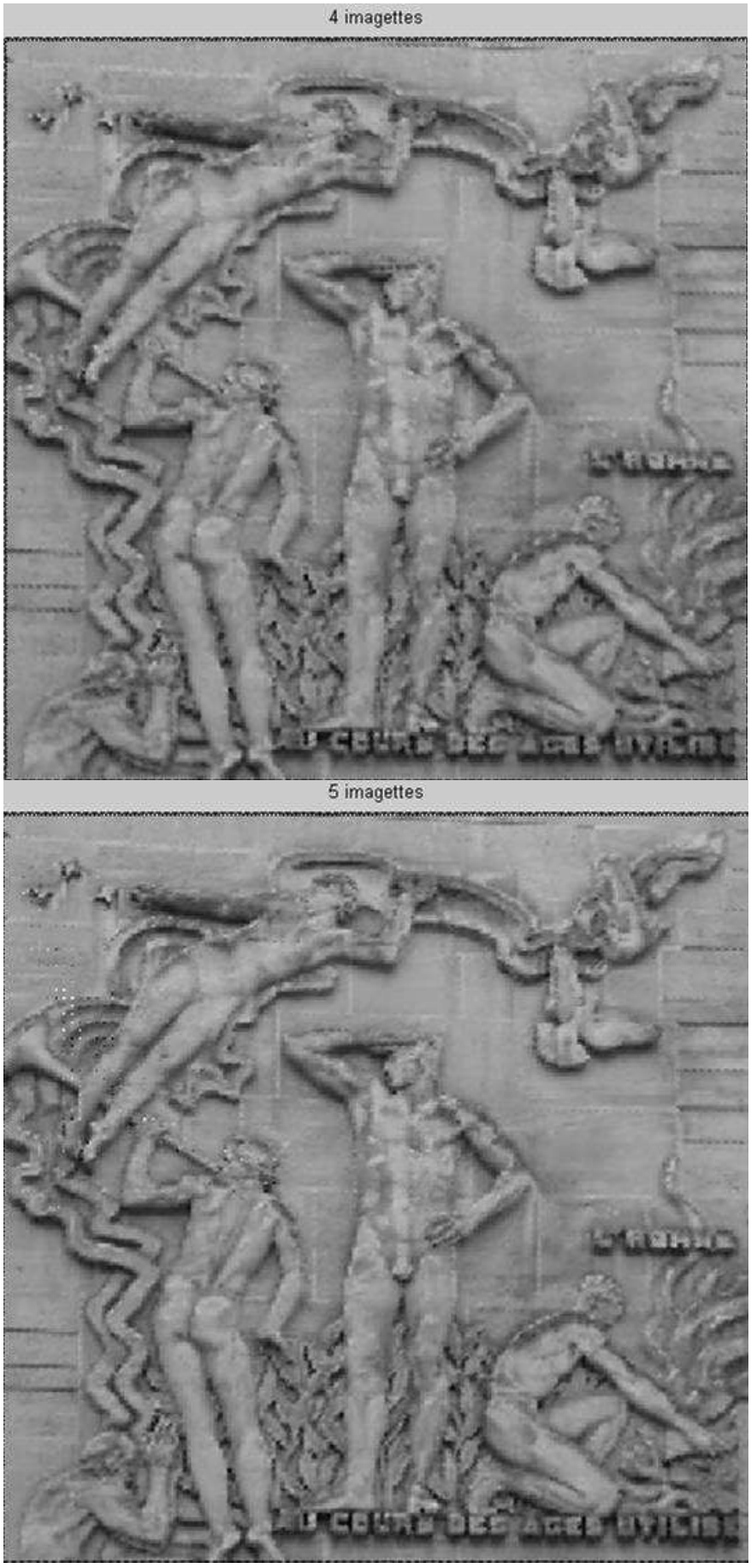
La dernière image que nous avons utilisée pour nos simulations a été prise avec une camera numérique à l'entrée de Telecom Paris. On a décidé de travailler avec un morceau de cette image de taille 375x375 pixels.

Voici l'image « haute résolution » originale et les reconstructions successives:

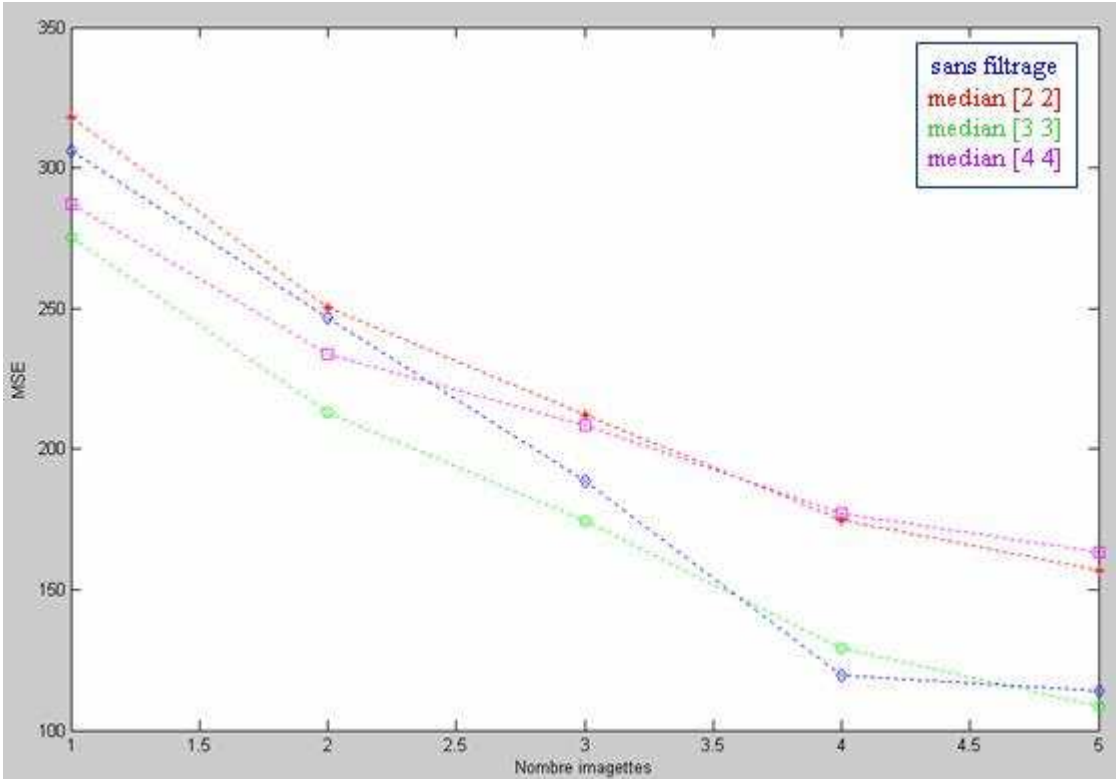


1 imagette


2 imagettes
3 imagettes


4 imagettes
5 imagettes

Et voici les résultats avec et sans filtrage médian de l'image reconstruite au fur et à mesure de l'ajout d'imagettes dans la reconstruction:



Discussion

La procédure pour la création d'image de haute résolution à partir d'images de basse résolution est très efficace de part plusieurs aspects :

- Cette procédure peut être facilement insérée à des implémentations hardware spécifiques permettant d'optimiser les calculs.

- Elle peut également être combinée à d'autres modules comme le filtrage d'images bruitées afin de rendre la résolution plus grande. On peut penser, par exemple, à filtrer les imagettes individuellement avant leur insertion et à utiliser aussi un filtre post-insertion pour l'image reconstruite (comme on a fait dans la section antérieure).

- L'algorithme d'insertion évite de reconstruire la triangulation de Delaunay dans son intégralité en réalisant une mise à jour locale de la triangulation. Grâce à cet algorithme, la procédure est bien adaptée à des applications en temps réel.

- On remarque aussi qu'il est aisé de généraliser cette procédure à des reconstructions d'image de haute résolution de dimensions plus élevées.

Au cours du travail que l'on a réalisé sur cette étude, on a également observé des aspects critiquables :

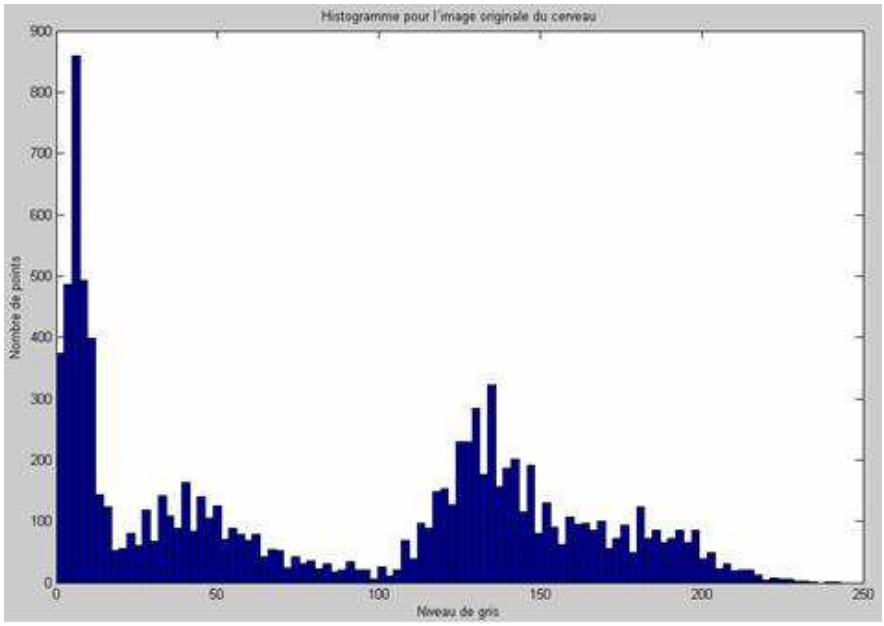
- Tout d'abord, l'apparition de quelques points «bruités» que l'on a appelé « bruit de reconstruction » et qui proviennent d'un calcul trop léger pour les surfaces d'approximation des triangles. Le nombre et distribution de ces points dépendent fortement du schéma d'échantillonnage choisi, comme on a montré.

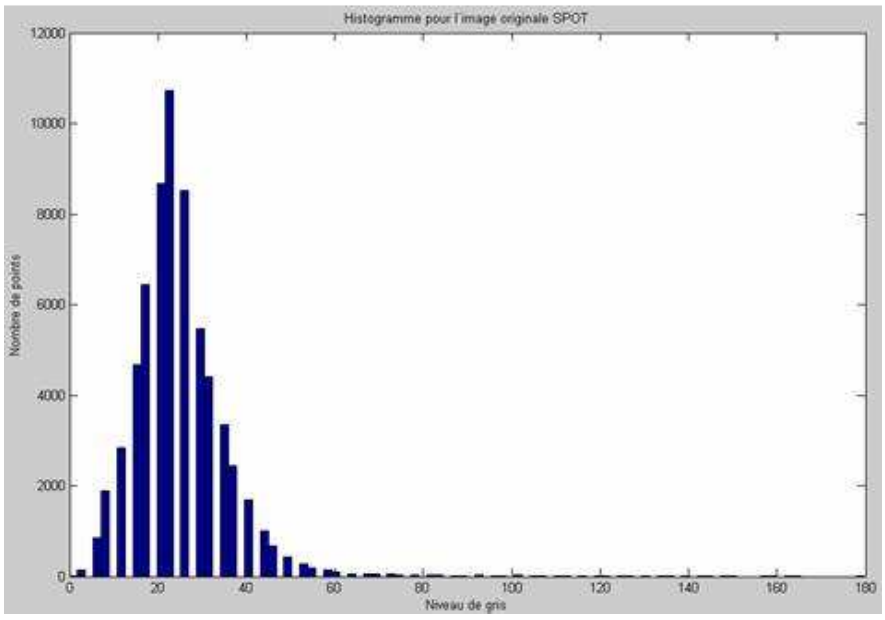
- On observe une certaine lenteur dans l'exécution de l'algorithme sur Matlab due à une complexité qui croit avec le nombre de triangles (création et élargissement de tables de coefficients et gradients, taille de la triangulation et recherches du triangle contenant un certain point, etc…) . On pense que cette obstacle devrait être relativement facile à surmonter sur une machine avec un hardware adapté au problème.

Au cours de nos expériences sur des images réelles, on a constaté que l'utilisation d'un filtre médian de fenêtre [3,3] pour l'image reconstruite apportait une amélioration évidente à l'image finale en terme de MSE.

On a également constaté que l'algorithme s'adaptait mieux à certaines types d'images, puisque, pour l'image SPOT on calcule à un MSE proche de 15, alors que pour l'image médicale du cerveau le MSE atteint des valeurs supérieures à 200 (avec 5 imagettes et le même schéma d'échantillonnage dans les deux cas**).**

Cette différence de comportement proviendrait peut être de la distribution des valeurs des pixels de l'image à reconstruire. Dans l'image SPOT, ces valeurs se concentrent sur un intervalle beaucoup plus petit que les valeurs de l'image médicale, comme on peut l'observer dans leurs histogrammes**:**





Bien que les valeurs niveaux de gris de l'image satellitaire aient été normalisées entre 1 et 255 (pour l'image complète, avant d'en prendre un morceau de 256x256 pixels—voir *script_SPOT.m*), ses valeurs sont beaucoup plus proches les unes des autres que les valeurs de l'image médicale, et il serait donc plus facile d'approximer les surfaces associées aux triangles obtenus pour l'image SPOT (on travaille sur un intervalle plus petit d'hauteurs sur la triangulation 3D, et donc il est plus facile d'obtenir des surfaces bien adaptées). C'est pourquoi l'erreur moyenne obtenue est décidément plus petite**.**

En tout cas, à l'avenir il serait intéressant de poursuivre les travaux avec la réalisation de nouveaux tests sur d'autres images et l'étude approfondie des résultats obtenus.