PROJET : MÉTHODE NUMÉRIQUE POUR L'INGÉNIEUR

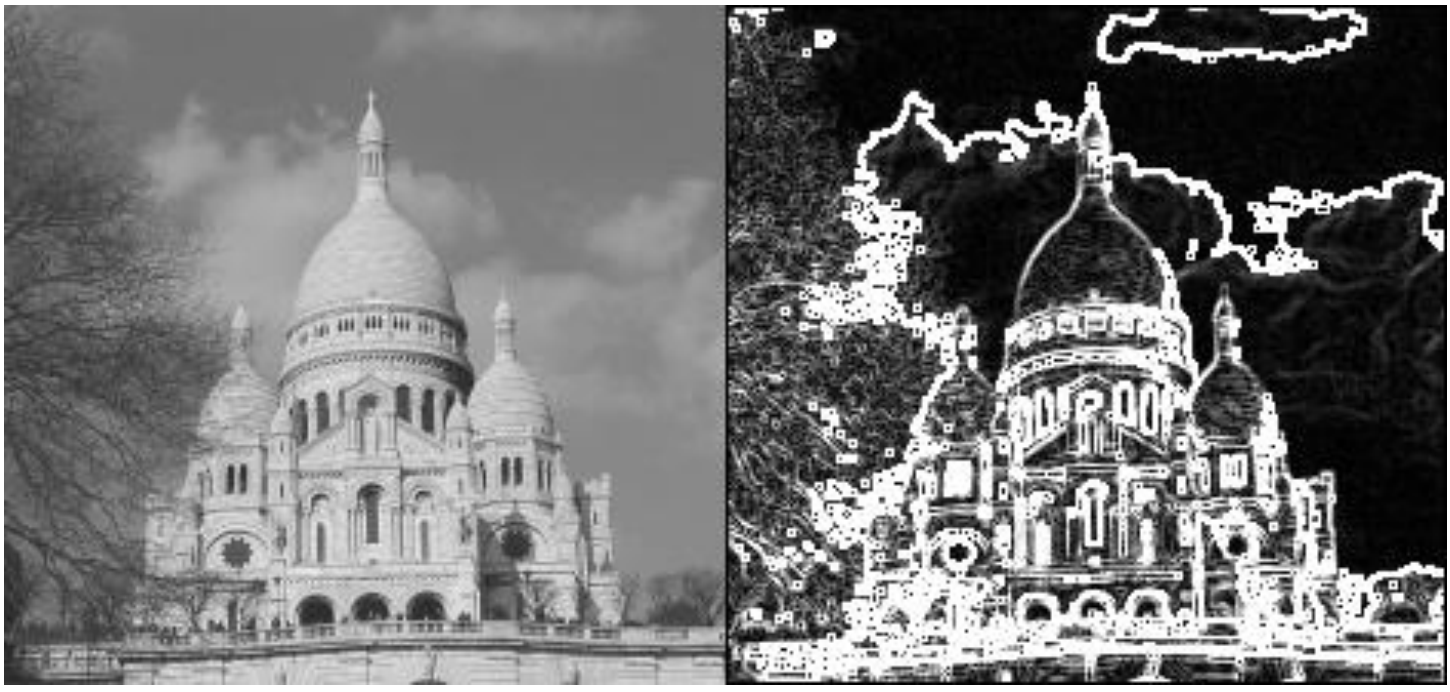

Figure 1 : Sacré-Cœur (gauche), Sacré-Cœur associé à un filtre de détection de contours (droite)

1. Objectif du projet

1.1. Contexte

Cette année, il nous a été expliqué le fonctionnement et les applications possible du Laplacien. Il nous a été donné lors des divers cours et séances de travaux dirigés des possibilités, des idées de systèmes sur lesquels il serait intéressant d'appliquer le Laplacien. Nous nous souviendrons des applications en acoustique pour étudier le son produit par un concert, ou encore la diffusion d'ondes pour une échographie, ou même l'évolution d'un gradient de température.

Une question légitime à se poser tout d'abord est : qu'est-ce que le Laplacien ? Le Laplacien est un outil mathématique. Mais pour commencer, nous nous devons de parler d'un autre outil mathématique très connu Nabla représenté par $\vec{\nabla}$ en France. Cet outil n'est pas très connu, il est plus couramment utilisé sous d'autre forme, en effet nous le connaissons plus avec les appellations *Grad, Div* et *Rot*, nous rappellerons que :

$$\rightarrow \overrightarrow{grad}\,\vec{u} = \vec{\nabla} \times \vec{u}$$

$$\rightarrow \overrightarrow{div}\,\vec{u} = \vec{\nabla} \cdot \vec{u}$$

$$\rightarrow \overrightarrow{rot}\,\vec{u} = \vec{\nabla} \wedge \vec{u}$$

Ces opérateurs sont utilisés dans de nombreux domaines de la physique comme de la mécanique des fluides. De ces trois opérateurs mathématiques en découle le Laplacien. En effet ce dernier, est défini comme le *Div* du *gradient*. Pour s'en convaincre nous pouvons faire la démonstration suivante :

$$\vec{\Delta}\,\vec{u} = \overrightarrow{div}\,(\,\overrightarrow{grad}\,\vec{u}\,) = \overrightarrow{div}\begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial z} \end{pmatrix} = \frac{\partial}{\partial x}\frac{\partial \vec{u}}{\partial x} + \frac{\partial}{\partial x}\frac{\partial \vec{u}}{\partial x} + \frac{\partial}{\partial z}\frac{\partial \vec{u}}{\partial z} = \frac{\partial^2 \vec{u}}{\partial x^2} + \frac{\partial^2 \vec{u}}{\partial y^2} + \frac{\partial^2 \vec{u}}{\partial z^2}$$

L'objectif de notre étude consistera à résoudre une équation issue de l'utilisation du Laplacien. Pour cela nous utiliserons la méthode des différences finies. Pour ce projet nous avons choisi de travailler sur le traitement d'image. Et plus particulièrement sur la détection de contours d'images.

Avant de rentrer de manière plus profonde dans notre sujet, parlons du filtre de détection de contours. Il prend en entrée une image est renvoie une image, mais celle-ci ne fait apparaitre que les contours des objets de l'image. Pour imager, sur la page de garde de ce document, vous y verrez une photographie du Sacré-Cœur à gauche. A droite, cette même photographie une fois le filtre de détection de contours appliqué. Le filtre de détection des contours est basé sur le principe de variation de fonction. En effet, sur l'image est repérée par des pixels blancs, les lieux où il y a de fortes variations d'intensité lumineuse. Nous essaierons d'appliquer le même principe en utilisant le Laplacien, car comme nous l'avons vu le Laplacien exprime une variation spatiale.

La détection de contours est une composante importante du traitement d'image puisqu'elle permet de reconnaître des objets présents dans une image. Elle peut aussi permettre de différencier les zones d'une image, voire de segmenter une image (distinguer les éléments de premier plan, second plan…). Cela peut ainsi permettre d'extraire un objet particulier d'une image.

Nous alors donc travaillé sur des images, nous nous accorderons sur l'importance de la définition d'une image. Une image est composée d'un ensemble de pixels. L'image possède donc, une longueur et un largueur. Ces deux distances ne sont pas mesurées grâce au système métrique, mais grâce à une autre unité. En effet, nous utiliserons le pixel comme unité de longueur. Un pixel est un atome de l'image, celui-ci est de taille finie et identique pour toute les images. De plus, un pixel décrit la couleur en ce point étudié grâce à la synthèse additive. Donc, chaque pixel est constitué de 3 canaux représentés par une valeur d'intensité lumineuse : une pour la couleur rouges vert et bleu.

1.2. Résultat attendu

Nous nous attendons à pouvoir rendre un programme sous Matlab, qui sera divisé en plusieurs parties :

- Une première partie sur la conversion de l'image en matrice dont les valeurs d'intensité de chaque canal pourront être exploitées par Matlab,
- Une seconde partie sur l'application du Laplacien sur la matrice extraite précédemment,
- Enfin, une troisième partie sur la transformation de la matrice finale en image qui sera affiche par Matlab.

3. Méthode numérique

3.1. Fonctionnement et application de l'algorithme de détection de contours

```
clear

Irgb = imread('Paysage.jpg');
Length = size(Irgb,1);
Width = size(Irgb,2);
I = rgb2gray(Irgb);
n=1; % Seuil de comptabilisation des pixels

% Filtrage moyenneur (lisseur) et normalisation

moyenneur = ones(3, 3) / 9;
I = conv2(I, moyenneur, 'same');

for C = 1:Length
  for L = 1:Width
    if I(C,L) > 255/2
      I(C,L)=255;
    else
      I(C,L)=0;
    end
  end
end
```

```
% Application du laplacien : différentielle finie centrée
Edge = [Length, Width];

for C = 1:Length
    for L = 1:Width
        if (C <= n) || (L <= n) % Conditions aux limites de Dirichlet/Neumann
            Edge(C,L)=0;
        elseif (C >= Length-n) || (L >= Width-n) % Conditions aux limites de Dirichlet/Neumann
            Edge(C,L)=0;
        else
            Dx2 = I(C+n,L) + I(C-n,L) - 2*I(C,L); % Approximation de la dérivée seconde selon x
            Dy2 = I(C,L+n) + I(C,L-n) - 2*I(C,L); % Approximation de la dérivée seconde selon y
            Dxy = I(C+n,L+n) + I(C-n,L-n) - 2*I(C,L); % Approximation de la dérivée seconde selon x et y
            Dyx = I(C-n,L+n) + I(C+n,L-n) - 2*I(C,L); % Approximation de la dérivée seconde selon y et x

            LP = Dx2 + Dy2 + Dxy + Dyx;
            % Ou encore :
            % LP = (I(C+n,L)+I(C-n,L)+I(C,L+n)+I(C,L-n)+I(C+n,L+n)+I(C-n,L-n)+I(C-n,L+n)+I(C+n,L-n))-8*I(C,L);

            if abs(LP/(n^2)) > 0 % Si passage par 0 alors contour détecté
                Edge(C,L)=255;
            else
                Edge(C,L)=0;
            end
        end
    end
end
```

```
subplot(1,3,1); % Affiche l'image originale
imshow(Irgb)
axis image

subplot(1,3,2); % Affiche l'image après le filtre moyenneur et normalisation
imagesc(I)
axis image

subplot(1,3,3); % Affiche l'image après application du laplacien
imshow(Edge)
colormap(gray(2))
axis image
```

Figure 2 : Code de détection de contours, Matlab

Afin d'élaborer un algorithme capable de détecter les contours d'une image, nous avons donc utilisé Matlab pour réaliser ce projet : il est déposé ci-dessus. Tout d'abord, l'algorithme reçoit une image en paramètre qu'il convertit en matrice à dimensions dont les deux premières représentent chaque pixel de l'image et dont la troisième, ayant une taille de 3, représente les valeurs RVB (Rouge, Vert et Bleu) de 0 à 255 du pixel considéré. Ensuite, nous avons transformé l'image en nuances de gris afin que la matrice soit en 2 dimensions. Un pixel est alors composé d'une seule valeur comprise entre 0 et 255.

Dans un second temps, sachant que le Laplacien est très sensible au bruit, nous allons appliquer un filtre moyenneur (une matrice 3x3 dont toutes les composantes sont égales à 1/9) afin de lisser l'image au maximum. L'objectif premier de ce filtre est d'éliminer un maximum de bruit qui sera susceptible de créer des erreurs dans l'application du Laplacien. On a alors une image presque en noir et blanc mais avec quelques nuances de gris visibles sur la matrice de l'image. On normalise alors l'image pour que les valeurs des pixels supérieurs à 255/2=127,5 soient blanches (valeur à 255) et les autres à noir (valeur à 0).

Dans un troisième temps, par l'intermédiaire d'une boucle, on parcourt chaque pixel de l'image afin de lui appliquer le Laplacien, en particulier la méthode par des différences finies centrées qui donne une approximation de la dérivée partielle d'ordre 2 de la fonction du pixel considéré :

- Pour chaque pixel contenu sur un bord de l'image, on applique les conditions aux limites de Dirichlet et Neumann, c'est-à-dire, en considérant systématique la valeur du pixel à 0 donc en noir.
- Pour tous les autres pixels, on applique les différences finies centrées à l'horizontale, la verticale et aux deux diagonales du pixel considéré qui nous donne donc une approximation de la dérivée seconde selon les 4 directions au pixel choisi, soit sa variation. La formule générale appliquée sur chaque pixel est la suivante :

$$\Delta I = \frac{I_{C+n,L} + I_{C-n,L} + I_{C,L+n} + I_{C,L-n} + I_{C+n,L+n} + I_{C-n,L-n} + I_{C-n,L+n} + I_{C+n,L-n} - 8.I_{C,L}}{n^2}$$

  Si la valeur absolue de la somme des dérivées secondes est positive cela signifie qu'il y a eu un passage de la fonction par 0 donc nous sommes en présence d'un contour. Le pixel se verra devenir blanc donc appliqué la valeur 255 et dans le cas contraire 0 pour le noir.

Finalement, on affiche sur une figure l'image originale, l'image filtrée par moyenneur et normalisée puis l'image après application du Laplacien. Les exemples ci-dessus illustrent les résultats de l'algorithme pour différentes images :

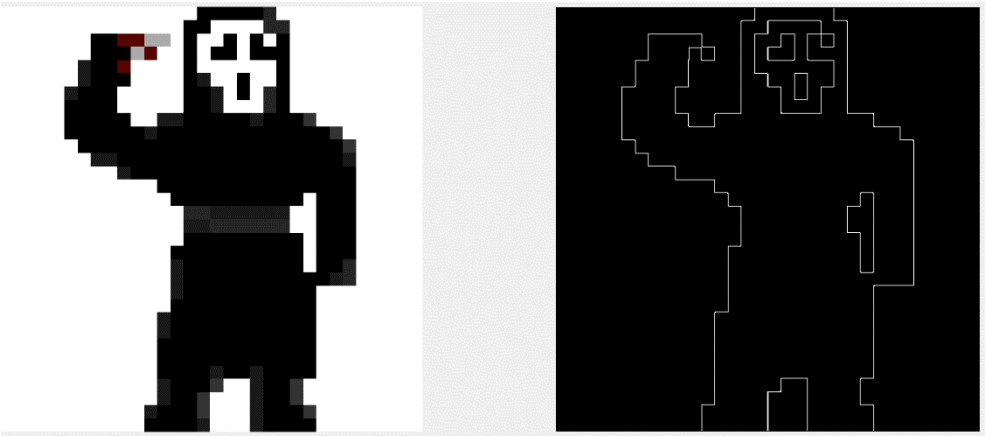

Figure 3 : Détection de contours, Ghostface_pixelart

Figure 4 : Détection de contours, Paysage

Figure 5 : Détection de contours, Blason_B12

Nous avons donc testé notre programme sur différentes images telles que des images pixels (pixel art), des paysages et des images avec des tailles conséquentes (9920x7016 pixels). On constate que les filtres transmettent fidèlement les contours principaux de chacune d'elles. Pour la troisième image, on pourrait penser que les contours transmis ne sont pas continus (n'apparaissent pas totalement) or ce n'est pas le cas. Ce phénomène est dû au rapport de taille entre un contour d'épaisseur un pixel et la taille de l'image. En zoomant sur celle-ci, on observe bien, que ces derniers sont continus :

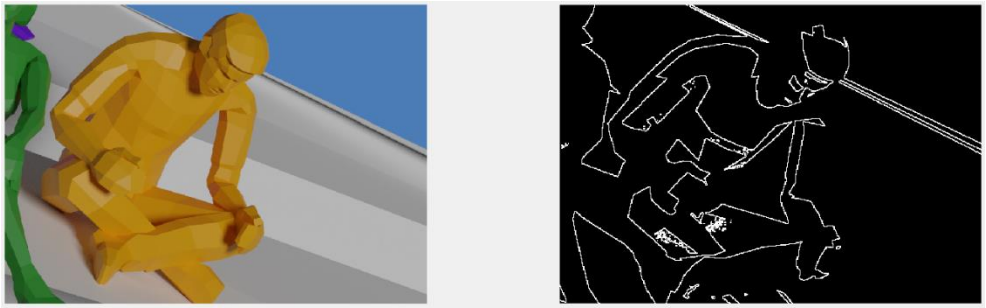

Figure 6 : Détection de contours, Blason_B12 zoom

On peut aussi noter que, selon la taille de l'image l'application du filtre moyenneur et du laplacien, le temps d'exécution du code est plus ou moins long. Cela est normal puisque le filtre moyenneur et laplacien effectuent des opérations sur chacun des pixels de l'images donc le nombre d'opérations est égal au nombre de pixels de celle-ci. Nous avons regroupé ces temps dans le tableau ci-dessous :

Image	Taille	Nombre d'opérations	Temps d'exécution	Opération/s
Paysage	282x425	119 850	0,11 s	1 089 545
Ghostface_pixelart	1024x1024	1 048 576	0,38 s	2 759 410
Blason_B12	9920x7016	69 598 720	2 min 10 s	535 375

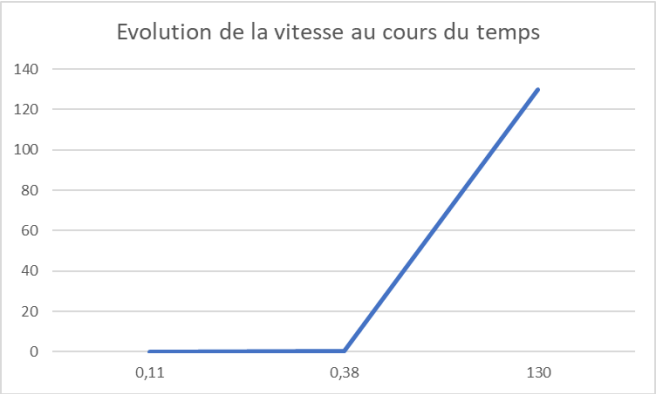

Figure 7 : Evolution de la vitesse au cours du temps

Nous observons que la vitesse n'est ni linéaire, ni constante. Nous tenterons d'expliquer ce phénomène par le calcul de la complexité de notre programme. Entre les lignes 14 et 22 de notre programme, nous obtenons une complexité *O(2 x C x L)*. En effet, nous observons deux boucles de C et L répétitions effectuant deux opérations. C'est équivalent à $O(n^2)$, soit une complexité quadratique.

Nous nous intéressons maintenant à la complexité des lignes 27 à 50. Sur une réflexion similaire, nous obtenons une complexité *O(2 x L x C)* dans le meilleur des cas, soit la première condition et une complexité *O(11 x L x C)*, dans le second. Nous obtenons donc, dans ces deux cas une complexité $O(n^2)$.

La somme des deux sera toujours une vitesse quadratique. Cela explique donc la différence entre les vitesses obtenues précédemment.

Nous avons ensuite tenté de faire parier le paramètre n et le coefficient du filtre moyenneur afin de savoir si notre système était le plus optimisé possible ou s'il était possible d'améliorer encore le filtre. Nous avons obtenu les résultats suivants :

- Si n augmente (> 1), on observe alors une diminution de la qualité de l'image.
- Si on augmente ou diminue le coefficient du filtre moyenneur (< 1/9 ou > 1/10) alors on perd de l'information lors de ce filtrage. Par conséquent, des contours disparaissent et la retranscription n'est pas maximale.
- Si le coefficient est compris entre 1/9 inclus et 1/10 inclus, la qualité et les informations de l'image sont conservées.

3.2. Avantages et inconvénients de notre algorithme

En comparant notre programme utilisant un filtre moyenneur et le Laplacien avec d'autres algorithmes tel que la méthode de détection de contours de Sobel ou encore la méthode du gradient qui n'effectue que les dérivées premières. Nous avons obtenu les résultats suivants :

Figure 8 : Basilique du Sacré-Cœur, méthode du laplacien puis de Sobel

Figure 9 : Photographe, méthode du laplacien puis du gradient

On peut remarquer, sur les images des photographes, qu'elles sont assez similaires. Néanmoins, dans notre cas aucun bruit n'est présent dans le ciel ce qui n'est pas le cas de l'image par le gradient lorsqu'on l'observe de plus près. On retrouve dans les deux cas des bruits au sol mais qui sont plus marqués dans notre cas en raison de la valeur des pixels à 255 tandis qu'elle est inférieure sur l'autre image (plus proche du gris). De plus, les détails de l'arrière-plan sont moins détaillés ce qui rend la distinction des bâtiments moins évidente.

Par rapport à la méthode de Sobel, (Basilique du Sacré-Cœur) là encore, dans notre cas, les détails sont moins retranscrits cependant les bruits sont totalement supprimés notamment au niveau des nuages.

Pour conclure, notre algorithme retranscrit très bien les formes de premier plan et les bruits sont totalement supprimés par le filtre moyenneur (lisseur) : les contours sont nets, d'une taille d'un pixel. Néanmoins, c'est ce même filtre qui lui fait défaut. En effet, sa forte influence sur le bruit dégrade aussi une partie de l'information utile ce qui provoque une perte de détails sur les éléments de second plan ou les petits objets.

4. Applications

Nous avons relevé plusieurs applications pour la détection de contours. Tout d'abord, cette méthode peut être et est utilisé pour la détection de formes. En effet, le fait de pouvoir afficher les contours d'un objet, permet ensuite à une intelligence artificielle de déterminer la catégorie de l'objet. Cette application est utilisée en imagerie médicale, mais aussi pour les reliefs en cartographie, en exploitant des images satellites mais aussi en suivit d'objet par l'intermédiaire d'une variante : la détection de contour actif.

4.1. Imagerie médicale

La détection de contours est beaucoup utilisée en imagerie médicale notamment la détection de contours actifs. Cette méthode repose sur la détection de contours classique mais incluant une composante évolutive. En effet, afin d'obtenir un contour stable épousant le plus possible la forme à détecter, les différents points formant le contour se déplacent afin d'affiner le résultat. De même, des données génétiques et chromosomiques sont exploitées dans ce type de méthode. L'exemple le plus courant dans le domaine médical est la détection du cancer du sein chez la femme par mammographie. Les différentes étapes effectuées sont les suivantes :

- Détection de contours de la glande mammaire par :
    - Seuillage,
    - Calcul de gradient,
    - Binarisation,
    - Initialisation du contour actif.
- Segmentation du muscle pectoral :
    - Binarisation,
    - Extraction du muscle.
- Détection des contours de l'anomalie avec un contour actif génétique.

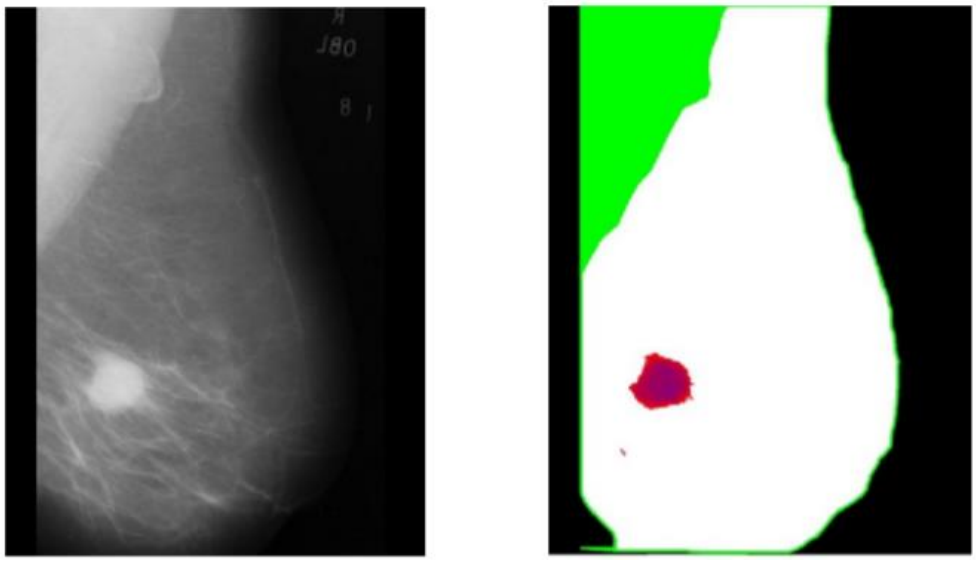

Figure 10 : Mammographie par détection de contours actifs et segmentation

Le principe de la segmentation sera définition de la partie suivante.

4.2. Segmentation d'images

Une deuxième application de la détection de contours est la segmentation d'images. Cela consiste en segmentant l'image en différents plans. Cette méthode regroupe les pixels similaires selon des critères d'intensité, de couleur etc. Elle permet ensuite de déterminer des régions (plans) dans l'image et de les évaluer en fonctions de la proximité par exemple. Cela peut être utile pour des voitures autonomes et pour la gestion du freinage en fonction de la proximité des autres acteurs de la route (véhicules, piétons, obstacles).

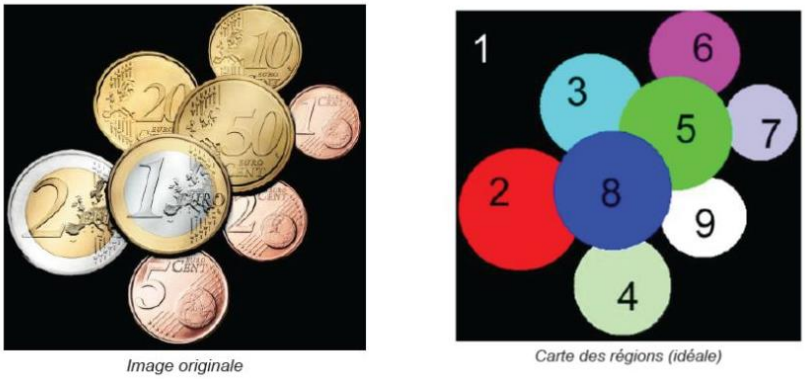

Figure 11 : Segmentation d'image

4.3. Détection de formes et modélisation 3D

En prenant différents points de vue d'un relief ou d'un corps, la détection de contours permet de mettre en surbrillance les formes et de reconstituer l'objet en 3D. Cette méthode est donc appliquée pour la modélisation 3D et permet d'obtenir des images ou cartographies en relief à partir de plusieurs images en deux dimensions.

Figure 12 : Programme Matlab appliqué à un panneau routier

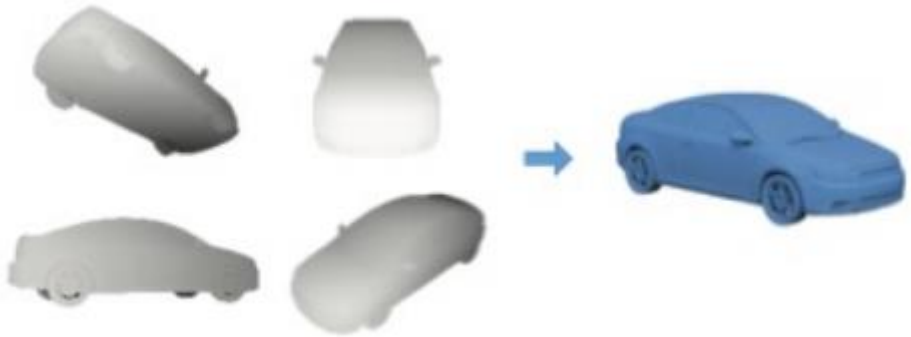

Figure 13 : Reconnaissance de formes

Nous pouvons également retrouver cette détection de contours dans la détection de présence ou encore de visages, notamment pour les caméras de surveillances. Cette méthode permet, dans ce cas, de déterminer l'identité d'une personne en réutilisant la détection de formes. Cette méthode est donc utilisée dans les domaines de la vidéo-surveillance et de la biométrie.

5. Conclusion

Pour conclure, de manière générale, aucun algorithme robuste pour la détection de contours n'a encore été trouvé malgré sa fréquente utilisation dans de nombreux domaines tels que l'imagerie médicale, la modélisation 3D ou la surveillance est sous de nombreuses variantes : différents filtres, différentes méthodes de détection… C'est un principe plutôt simple à mettre en œuvre et intuitif pour sa compréhension. Comme nous avons pu le remarquer dans notre algorithme, par l'utilisation du laplacien (différences finies donnant une approximation des dérivées secondes), nous avons obtenu des résultats plus que satisfaisants avec des formes discernables et détaillées.

Le principal défaut de cette méthode mais qui est aussi le talon d'Achille de tous les algorithmes de détection de contours est la présence de bruits qui perturbe les performances de ces derniers. En effet, ils provoquent l'apparition de contours sur des formes inexistantes ainsi une baisse de la qualité des détails qui sont superposés aux bruits. Pour pallier ce problème, il est donc nécessaire d'applique un filtre qui va permettre de lisser l'image avant son traitement. Cela permet de réduire voire de supprimer le bruit dans certaines zones de l'image ainsi que d'obtenir des contours plus lissés. Revers de la médaille, la puissance du filtre peut causer un lissage trop important de l'image et par conséquent une perte d'informations à transmettre (contours incomplets ou n'apparaissant pas). On retrouve ce phénomène très prononcé dans les images homogènes en couleurs ou très sombres naturellement.

Un compromis très minution entre choix de filtre et de méthode est très important afin d'éliminer un maximum de bruit tout en conservant un maximum d'informations afin de retranscrire des contours d'une image nets, détaillés et continus.

6. Table des illustrations