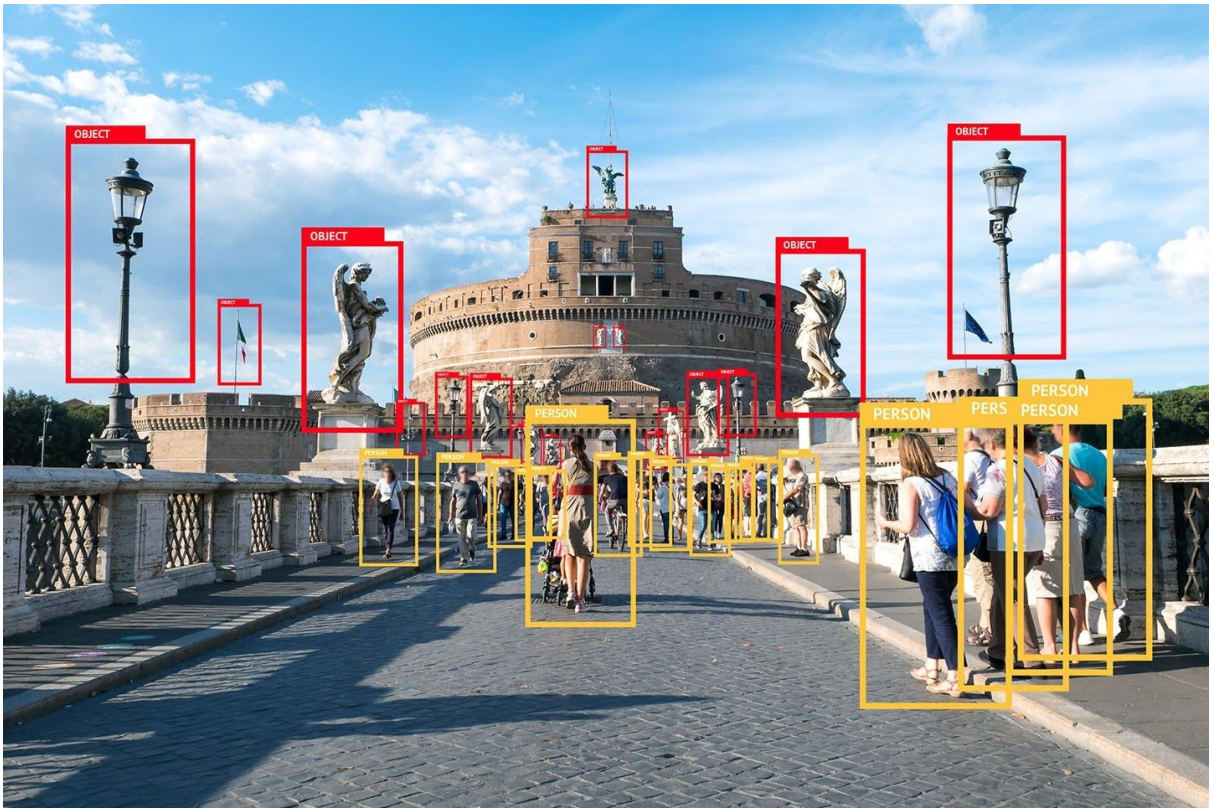
Reconnaissance d’objets

Traitement d’images et vision



Encadré par : Madame POREBSKI Alice

NGUYEN Martin
BONNET Quentin
LOUARD Abir

SOMMAIRE

INTRODUCTION

Dans le cadre de notre deuxième année d'études supérieures dans la spécialité informatique à EIL Côte d'Opale, nous avons eu l'opportunité d'avoir " La reconnaissance d'objets " comme projet de traitement d'images et vision.

Ce projet nous a permis d'appliquer des notions étudiées en cours, que ce soit des notions techniques de programmation (Matlab, Scilab) ou des notions organisationnelles.

La reconnaissance d'objets est une technique de "Computer vision" utilisée pour l'identification d'objets présents dans des images et des vidéos. Cette dernière s'avère être le produit d'algorithmes de Deep Learning et de Machine Learning. Lorsqu'un être humain observe des photos ou regarde une vidéo, il est en mesure de percevoir immédiatement les personnes, les objets, les scènes et les détails visuels qu'il a sous les yeux.

Le but de la reconnaissance d'objet est d'apprendre à un ordinateur à réaliser ce dont les humains sont naturellement capables, et à acquérir un niveau de compréhension approprié de ce que contient l'image.

REMERCIEMENTS

Nous tenons à remercier dans un premier temps, toute l'équipe pédagogique au sein de l'EIL Côte d'Opale qui assure la continuité des études en utilisant de différentes plateformes.

Nous sommes également reconnaissants à notre professeure Madame POREBSKI qui n'a pas cessé de nous aider et de nous donner des conseils concernant les missions évoquées dans ce rapport, qu'elle nous a apporté lors des différents suivis et la confiance qu'elle nous a témoigné.

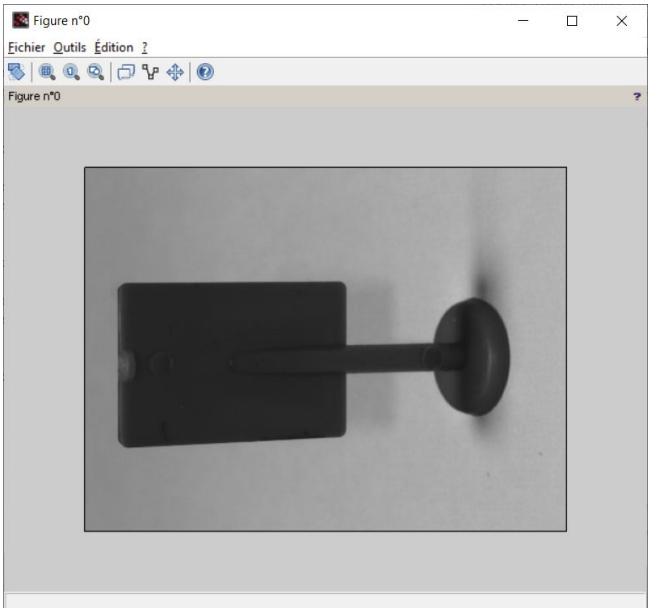
La reconnaissance d'objets a pour but d'identifier un objet quelconque dans une base de données image ou une vidéo. Cette identification est effectuée grâce à des attributs décrivant la forme, la couleur et la texture. Dans ce TP, nous avons 4 classes d'objets (panneaux de signalisation, personnages, voitures et camions). Pour chaque classe nous avons 6 objets différents et chaque objet sera représenté selon 5 angles de vue. Un objet de chaque classe sera utilisé afin de construire la base d'apprentissage, et les autres 5 objets de la classe seront utilisés pour tester la pertinence de la caractérisation. La base d'apprentissage est constituée de 20 images (5 images par classe) et la base à tester de 100 images (25 images par classe).
Ce TP est sous forme de 22 questions qui répondent à cette problématique. Chaque question a son script et son image correspondante.

Question n°1 :

- Création d'un nouveau fichier " TP.sce " , la lecture de l'image 000002.png et la conversion de cette dernière en niveaux gris.

script_TP.sce

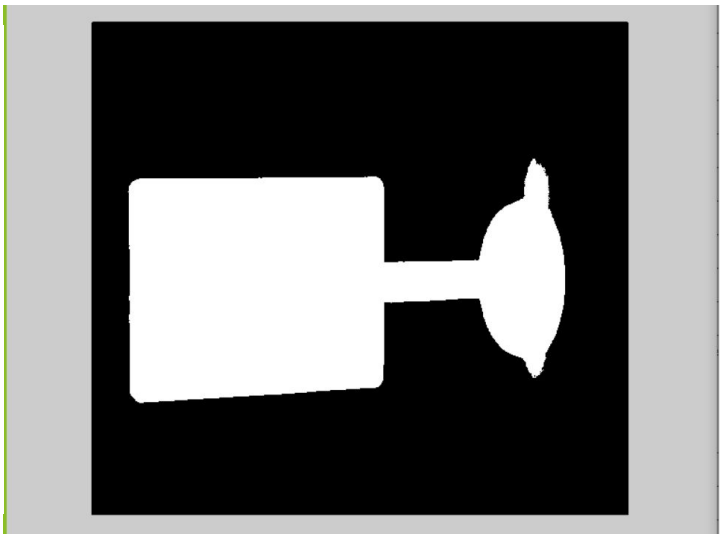
```
1 //·Question·1
2 I=imread('Base\000002.png');
3 G=rgb2gray(I);
4 figure;·imshow(G);
5
```



Question n°2 :

- Binarisation de l'image en niveau de gris (la forme en blanc et le fond en noir).

```
// Question 2

I=imread('Base\000002.png');
G=rgb2gray(I);
B =im2bw(G,0.5);
B2=imcomplement(B);
/*for i= 1:length(B(:,1))
    for j=1:length(B(1,:))
        if B(i,j)==-1 then
            B(i,j)=0;
        else B(i,j)=-1
        end
    end
end*/

figure; imshow(B2);

```

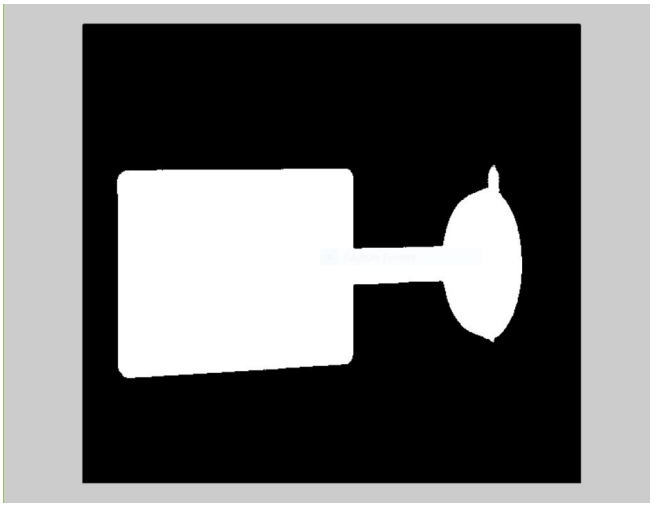


Par défaut la fonction “ *im2bw* ” donne la forme en noire et le fond en blanc, pour remédier à cet inconvénient on utilise la fonction “ *incomplement* ” qui pour chaque pixel change la valeur à false si c’est true et à true si c’est false.

Question n°3 :

- Le seuil utilisé pour binariser les images pouvant varier d’une image à l’autre, utiliser la fonction “ *imgraythresh* ” pour déterminer de manière automatique le seuil en fonction de l’image passée en paramètre.

```
// Question 3

I=imread('Base\000002.png');
G=rgb2gray(I);
seuil = imgraythresh(G);
B =im2bw(G,seuil);
B2=imcomplement(B);
figure; imshow(B2)

```

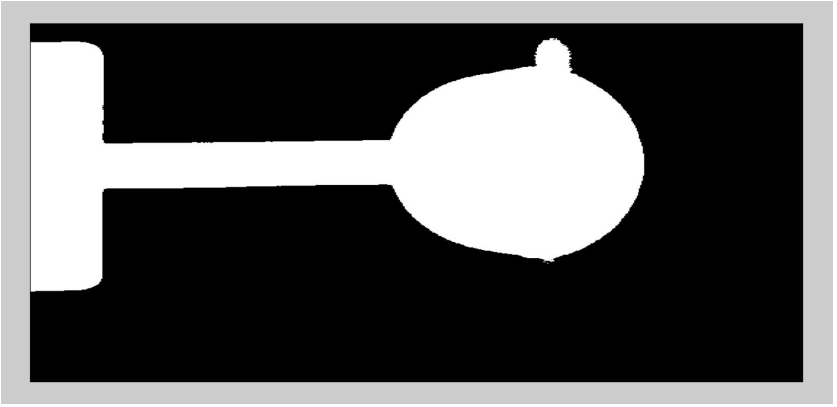


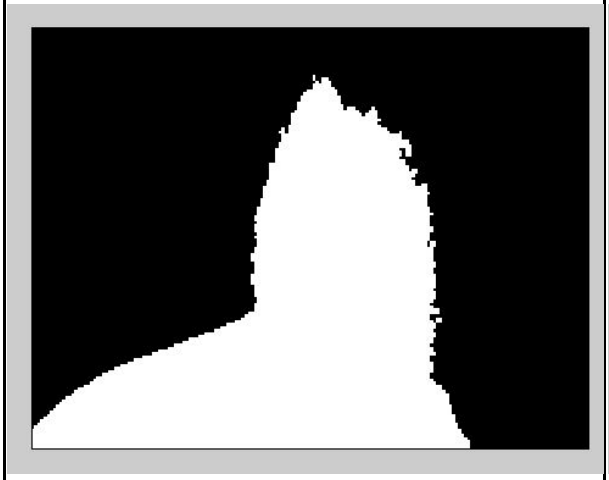
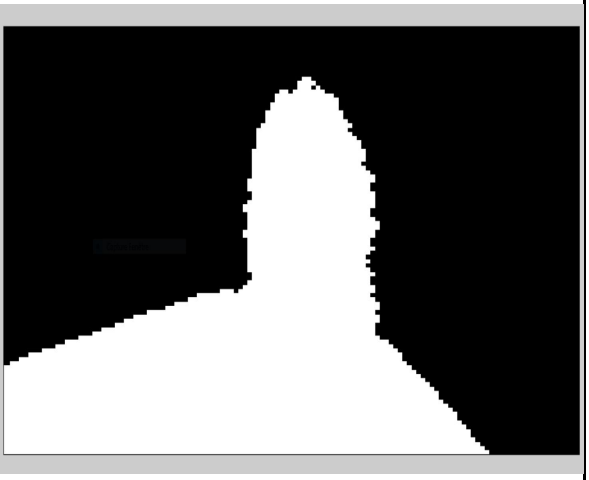
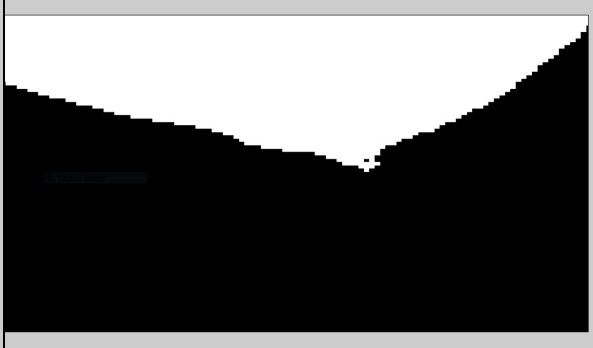
Question n°4 :

- Utiliser les fonctions morphologiques nécessaires pour améliorer votre binarisation. Vérifier en zoomant sur l’image obtenue qu’il ne reste pas de

petits pixels parasites sur le fond de l'image et que les pixels relatifs à l'objet appartiennent bien à une unique région.

```
34
35 // Question 4
36
37 I=imread('Base\000002.png');
38 G=rgb2gray(I);
39 seuil = imgraythresh(G);
40 B =im2bw(G,seuil);
41 B2=imcomplement(B);
42 e=imcreatese('cross',3,3);
43 Ir=imerode(B2,e);
44 Id=imdilate(Ir,e);
45 figure; imshow(Id);
46
47
```



Avant	Après
	
	

Ici, étant donné qu'on a des pixels parasites blancs sur fond noir, on effectue une ouverture, c'est à dire une érosion suivie d'une dilatation. Pour se faire on utilise un élément structurant en forme de croix de cette forme et taille :

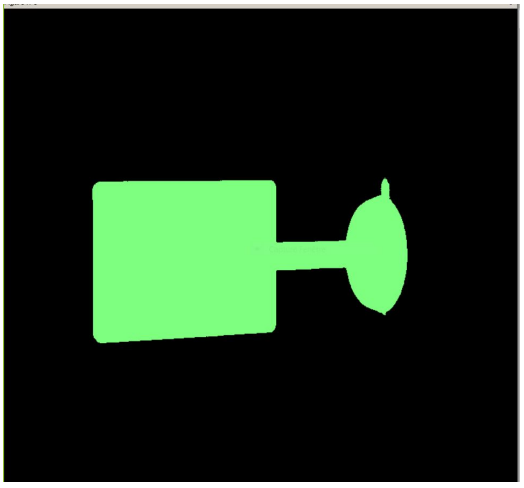
0	1	0
1	1	1
0	1	0

Cette méthode nous permet ici d'obtenir une forme composée d'une seule région, c'est-à-dire sans pixels blancs isolés.

Question n°5 :

- Vérifier que le nombre de régions présentes dans l'image est bien égal à 1. Si ce n'est pas le cas, réajuster vos opérations morphologiques.

```
48
49 //-Question-5
50
51 clear all;
52 clc;
53
54 I=imread('Base\000002.png');
55 G=rgb2gray(I);
56 seuil = imgraythresh(G);
57 B =im2bw(G,seuil);
58 B2=imcomplement(B);
59 e=imcreatese('cross',3,3);
60 Ir=imerode(B2,e);
61 Id=imdilate(Ir,e);
62 [Ii,r]=imlabel(Id);
63
64 figure; imshow(Ii,jetcolormap(r));
65
```



r	1	Double

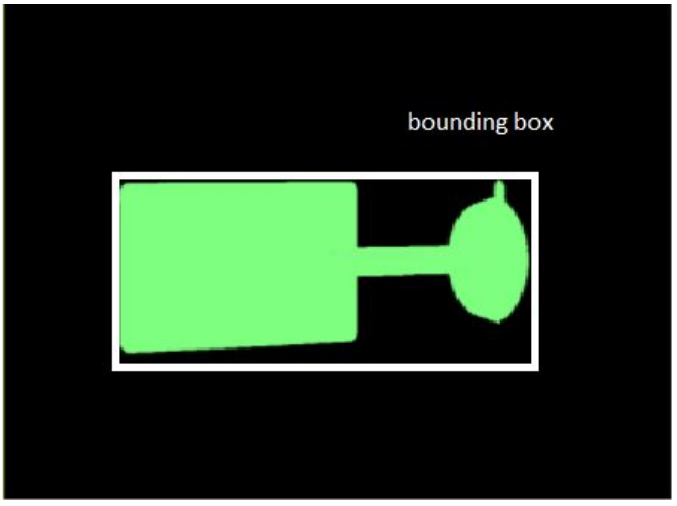
On observe avec la fonction " *jetcolormap* " la forme colorée d'une seule couleur, de plus la variable r représentant le nombre de régions de l'image est égale à 1 donc on est sûr à 100% qu'il n'y a qu'une seule région.

Question n°6 :

- Calculer les coordonnées du centre de gravité de la forme, ainsi que sa surface et les coordonnées du cadre circonscrit.

```
68 //·Question·6
69
70 I=imread('Base\000002.png');
71 G=rgb2gray(I);
72 seuil = imgraythresh(G);
73 B =im2bw(G,seuil);
74 B2=imcomplement(B);
75 e=imcreatese('cross',3,3);
76 Ir=imerode(B2,e);
77 Id=imdilate(Ir,e);
78 [Ii,r]=imlabel(Id);
79 [S,Bb,Cg]=imblobprop(Ii);
80 disp(S,Bb,Cg);
81
```

```
1.

346653.

89.
316.
1042.
475.

514.66832
545.58679
```



Surface	346653
Bounding-Box	x_{bb} = 89 L_{bb} = 1042 y_{bb} = 316 l_{bb} = 475
Coordonnées centre de gravité	x_g = 514,66832 y_g = 545,58679

Question n°7 :

- Déduire des coordonnées du cadre circonscrit la longueur et la largeur de la région.

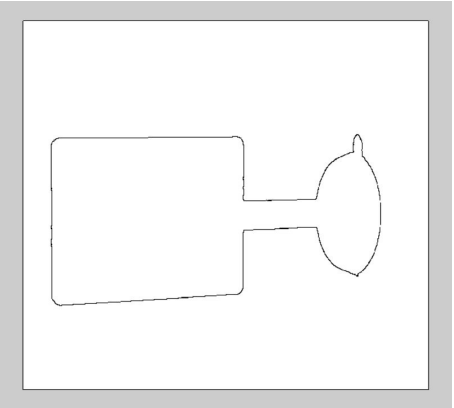
Coordonnées cadre circonscrit	x = 89	y = 316
Longueur et largeur de la région	L = 1043	l = 316

Grâce à la fonction " *imblobprop* " utilisée dans la question n°6, nous avons pu déterminer les coordonnées ainsi que la taille de la région.

Question n°8 :

- Le périmètre est également un attribut qu'il est intéressant de considérer pour caractériser la forme d'un objet. Afin de pouvoir le calculer, utiliser l'outil "*edge*" qui permet de mettre en évidence les contours dans une image.

```
100
101 // Question 8
102
103 I=imread('Base\000002.png');
104 G=rgb2gray(I);
105 seuil = imgraythresh(G);
106 B =im2bw(G,seuil);
107 B2=imcomplement(B);
108 e=imcreatese('cross',3,3);
109 Ir=imerode(B2,e);
110 Id=imdilate(Ir,e);
111 E=edge(Id,'prewitt');
112 E2=imcomplement(E);
113 figure; imshow(E2);
114
```



La fonction "*edge*" prend en entrée une image binarisée et elle nous retourne le contour de la forme, ce qui rend le calcul du périmètre relativement aisé.

Question n°9 :

- ❖ Une fois le contour déterminé, proposer une solution permettant de compter le nombre de pixels constituant le périmètre de la forme.

```
124
125 // Question 9
126
127
128 I=imread('Base\000002.png');
129 G=rgb2gray(I);
130 seuil = imgraythresh(G);
131 B =im2bw(G,seuil);
132 B2=imcomplement(B);
133 e=imcreatese('cross',3,3);
134 Ir=imerode(B2,e);
135 Id=imdilate(Ir,e);
136 E=edge(Id,'prewitt');
137 E2=imcomplement(E);
138 Per = sum(E==%T);
139 disp(Per);
140
141
```

```
Console Scilab 6.1.0

   7471.

-->
```

Ainsi pour calculer le périmètre, on compte le nombre de pixels qui forment le contour.

Question n°10 :

- ❖ Tester l'approche sur l'image 000041.png et compléter le code pour que l'extraction des attributs de reconnaissance de forme ne s'effectue que sur la région de l'image qui possède la plus grande aire.

```
I=imread('Base\000061.png');
G=rgb2gray(I);
seuil = imgraythresh(G);
B =im2bw(G,seuil);
B2=imcomplement(B);
e=imcreatese('cross',3,3);
Ir=imerode(B2,e);
Id=imdilate(Ir,e);
[Ii,r]=imlabel(Id);
[Si,Bbi,Cgi]= imblobprop(Ii);
```

Ii est une image indicée qui contient des valeurs entières de 0 à r avec r le nombre de régions.
r = nombre de régions
Si est le vecteur qui contient l'aire de chaque région
Idem pour la bounding box et le centre de gravité
Bbi est de dimension 4 x r
Cgi est de dimension 2 x r

```
[Maximum_aire,indice_max] = max(Si);

S=Si(indice_max);
Bb=Bbi(:,indice_max);
Cg=Cgi(:,indice_max);
figure; imshow(Ii);
```

Ici on prend les attributs de forme de la région qui a la plus grande aire, or, Si est le vecteur qui stocke les aires des régions de manière **ordonnée**. On récupère l'indice max et ainsi on obtient les attributes de reconnaissance de la région avec la plus grande aire.

On obtient :

Aire	Bounding box	Centre de gravité
4379668.0	189.0 206.0 759.0 628.0	587.7301616148845 517.3594166482295

Question n°11 :

- Écrire une fonction appelée " *AttributsForme.sci* " permettant de retourner le vecteur des attributs de forme (aire, périmètre, longueur et largeur de la forme) à partir d'une image en niveau de gris.

```
function [Aire,Perimetre,Longueur,Largeur]=AttributsForme(Img_gris)

    s=imgraythresh(Img_gris);
    Img_binaire = im2bw(Img_gris,s);
    Img_binaire = imcomplement(Img_binaire);
    e=imcreatese('cross',3,3);
    Ir=imerode(Img_binaire,e);
    Id=imdilate(Ir,e);
    E=edge(Id,'prewitt');

    [Ii,r]=imlabel(Id);
    [Si,Bbi,Cgi]= imblobprop(Ii);
    [max_aire,indice_max_aire] = max(Si);

    Aire = Si(indice_max_aire);
    Perimetre = sum(E==%T);
    if Bbi(3)>Bbi(4) then
        Largeur = Bbi(4);
        Longueur =Bbi(3);
    else
        Largeur = Bbi(3);
        Longueur =Bbi(4);
    end


endfunction
```

Ici on ne fait que réutiliser ce qu'on a vu précédemment.

Question n°12 :

- Tester cette fonction au sein du " *fichierTP.sci* ". Noter qu'il est nécessaire d'exécuter au préalable la commande *exec* " *AttributsForme.sci* " pour que la fonction puisse être prise en compte.

```
// Question 12

clear;

exec AttributsForme.sce;

I=imread('Base\000061.png');
G=rgb2gray(I);
[aire,perimetre,longueur,largeur] = AttributsForme(G);
```

Nom	Valeur	Type	Visibilité	Memory
G	1024x1280	Entier	local	1,3 MB
I	1024x1280x3	Entier	local	3,9 MB
aire	3.8e+05	Double	local	216 B
largeur	628	Double	local	216 B
longueur	759	Double	local	216 B
perimetre	2.96e+04	Double	local	216 B

On remarque qu'on obtient les mêmes valeurs qu'à la question 10 donc la fonction devrait bien fonctionner.

Question n°13 :

- ❖ Réaliser l'apprentissage du processus de classification.

```
exec AttributsForme.sce
nb_classe = 4;
nb_image =30;
nb_ima = nb_classe*nb_image;
nb_ima_train=nb_ima/6;
ima_label=0;
for i_train=1:nb_ima
    if(((i_train>=1)&&(i_train<=5))|||((i_train>=31)&&(i_train<=35))|||((i_train>=61)&&(i_train<=65))|||((i_train>=91)&&(i_train<=95))) then
        ima_label=ima_label+1;
        // Enregistrement du numéro de la classe dans un tableau
        num_classe(ima_label)=floor((i_train-1)/nb_image)+1;
        // Concaténation des chaînes de caractères
        // pour constituer le chemin d'accès au fichier image
        if(i_train/10 <1)
            fichier_train = strcat(['Base\00000',msprintf('%d',i_train),'.png']);
        else
            if(i_train/100<1)
                fichier_train = strcat(['Base\0000',msprintf('%d',i_train),'.png']);
            else
                fichier_train = strcat(['Base\000',msprintf('%d',i_train),'.png']);
            end
        end
        disp([fichier_train 'Classe' msprintf('%d',num_classe(ima_label))]);
        // ouverture de l'image
        Ima_train=imread(fichier_train);

        Ima_gray=rgb2gray(Ima_train);
        [aire,perimetre,longueur,largeur]=AttributsForme(Ima_gray);
        Attributs(ima_label,1)= aire;
        Attributs(ima_label,2)= perimetre;
        Attributs(ima_label,3)= longueur;
        Attributs(ima_label,4)= largeur;
    end
end
```

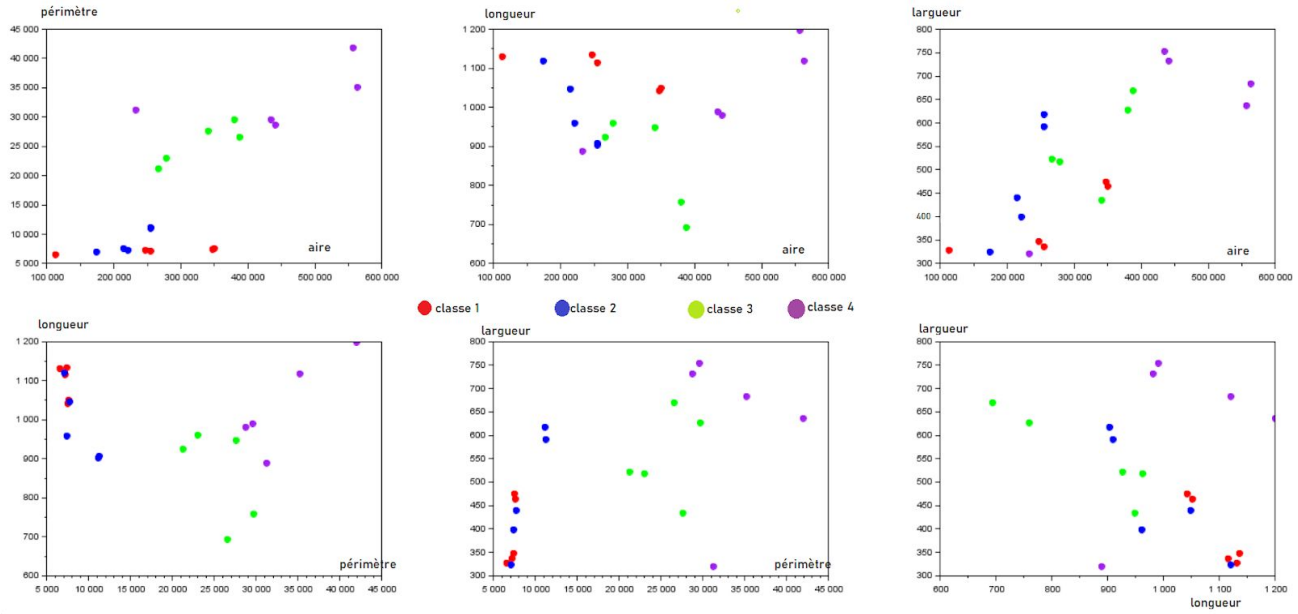
On obtient une matrice Attributs de taille 20 x 4 avec l'aire en première colonne, périmètre en deuxième, longueur en troisième et largeur en quatrième.

1	2	3	4
3,4878e+05	7,561e+03	1,05e+03	465
3,4665e+05	7,471e+03	1,042e+03	475
2,5403e+05	7,091e+03	1,115e+03	337
2,4637e+05	7,345e+03	1,135e+03	348
1,1188e+05	6,553e+03	1,131e+03	328
2,5399e+05	1,1117e+04	903	618
2,5442e+05	1,1189e+04	908	592
2,1986e+05	7,377e+03	960	399
2,1361e+05	7,61e+03	1,047e+03	441
1,7318e+05	7,026e+03	1,12e+03	325
3,7967e+05	2,963e+04	759	628
3,8706e+05	2,6541e+04	693	670
3,3975e+05	2,7598e+04	948	435
2,7815e+05	2,299e+04	961	518
2,6552e+05	2,116e+04	925	523
5,5596e+05	4,1932e+04	1,199e+03	637
5,6276e+05	3,5176e+04	1,119e+03	684
2,3228e+05	3,1203e+04	889	321
4,4113e+05	2,8661e+04	981	732
4,3346e+05	2,9568e+04	990	754

Question n°14 :

- Visualiser les nuages de points dont les coordonnées sont deux des attributs calculés. Déterminer ensuite visuellement les attributs qui semblent les plus pertinents pour la classification.

```
xdel(winsid());

tabi=[1 2;
      1 3;
      1 4;
      2 3;
      2 4;
      3 4];

for i=1:6
    subplot(2,3,i);

    p1=scatter(Attributs((1:5),tabi(i,1)),Attributs((1:5),tabi(i,2)));
    p2=scatter(Attributs((6:10),tabi(i,1)),Attributs((6:10),tabi(i,2)));
    p3=scatter(Attributs((11:15),tabi(i,1)),Attributs((11:15),tabi(i,2)));
    p4=scatter(Attributs((16:20),tabi(i,1)),Attributs((16:20),tabi(i,2)));

    p1.mark_background=color("red");
    p1.mark_foreground=color("red");
    p2.mark_background=color("blue");
    p2.mark_foreground=color("blue");
    p3.mark_background=color("green");
    p3.mark_foreground=color("green");
    p4.mark_background=color("purple");
    p4.mark_foreground=color("purple");

end
```



Visuellement, les paramètres qui paraissent les plus pertinents sont le périmètre et la longueur. Autrement dit la vignette en bas à gauche.

Question n°15 :

- Centrer et réduire les attributs de forme calculés pendant l'apprentissage.

```
for i=1:4
    moyenne=mean(Attributs(:,i));
    ecart_type=stdev(Attributs(:,i));
    Attributs_normalises(:,i)=(Attributs(:,i)-moyenne)./ecart_type;
end
```

1	2	3	4
0,2847	-0,9498	0,4356	-0,3211
0,2668	-0,9574	0,3736	-0,252
-0,5121	-0,9897	0,9389	-1,2049
-0,5765	-0,9681	1,0937	-1,1289
-1,7075	-1,0354	1,0627	-1,267
-0,5124	-0,6476	-0,7027	0,7354
-0,5088	-0,6415	-0,664	0,5558
-0,7994	-0,9654	-0,2613	-0,7768
-0,852	-0,9456	0,4123	-0,4868
-1,1919	-0,9952	0,9776	-1,2877
0,5444	0,9252	-1,8177	0,8044
0,6066	0,6628	-2,3287	1,0944
0,2087	0,7526	-0,3542	-0,5282
-0,3092	0,3611	-0,2536	0,0449
-0,4154	0,2056	-0,5323	0,0794
2,0269	1,9704	1,5893	0,8665
2,0841	1,3964	0,9698	1,1911
-0,695	1,0589	-0,8111	-1,3154
1,0613	0,8429	-0,0987	1,5225
0,9968	0,92	-0,029	1,6744

Question n°16 :

- ❖ Compléter le programme précédent afin d'ouvrir une image de la base test et classer cette image par l'algorithme du plus proche voisin.

```
function [classe]=classifieur(attributs_normalises,img_gris,moyenne,ecart_type,num_classe)
    [aire,perimetre,longueur,largeur]=AttributsForme(img_gris);
    res=[aire,perimetre,longueur,largeur];
    for i=1:4
        res(i)=(res(i)-moyenne(i))./ecart_type(i);
    end
    for app = 1:20
        //tableau_traitement=sqrt(abs(attributs_normalises(:,i)-res(i))^2);
        distance(app)=sum(abs(attributs_normalises(app,:)-res(1,:)));
        //distance(app)=sqrt(sum((attributs_normalises(app,:)-res(1,:)).^2));
    end
    [dist_mini,ind] = min(distance);
    classe=num_classe(ind);
endfunction
```

Nous avons décidé de créer une fonction qui se chargera d'attribuer à l'image en entrée une classe. Le principe est le suivant, on récupère les attributs de l'image, on les normalise et ensuite on calcule le vecteur de distances entre les attributs des images d'apprentissages et les attributs de l'image d'entrée. La distance utilisée est la distance de Manhattan. Enfin, on récupère la distance minimale, et son indice nous donne la classe estimée.

Question n°17 :

- ❖ Écrire un programme permettant de calculer le taux de classification en vous inspirant du programme permettant l'apprentissage. Un moyen de valider votre programme est de classer les images de la base d'apprentissage et de vérifier que vous obtenez bien un taux de 100%.

```
exec AttributsForme.sce
exec classifieur.sce
nb_classe = 4;
nb_image =30;
nb_ima = nb_classe*nb_image;
nb_ima_train=nb_ima/6;
reussite=0;
ima_label=0;
for i_test=1:nb_ima
    if(((i_test>=6)&&(i_test<=30))|||((i_test>=36)&&(i_test<=60))|||((i_test>=66)&&(i_test<=90))|||((i_test>=96)&&(i_test<=120))) then
        ima_label=ima_label+1;
        num_classe_test(ima_label)=floor((i_test-1)/nb_image)+1;
        if(i_test/10 <1)
            fichier_test = strcat(['Base\00000',msprintf('%d',i_test),'.png']);
        else
            if(i_test/100<1)
                fichier_test = strcat(['Base\0000',msprintf('%d',i_test),'.png']);
            else
                fichier_test = strcat(['Base\000',msprintf('%d',i_test),'.png']);
            end
        end
        Ima_test=imread(fichier_test);
        Ima_gray=rgb2gray(Ima_test);
        classe_estimee=classifieur(Attributs_normalises,Ima_gray,moyenne,ecart_type,num_classe);
        if(classe_estimee==num_classe_test(ima_label))
            reussite=reussite+1;
        end
    end
end
taux=(reussite*100)/100;
```

En exécutant ce code, on classe toutes les images, exceptées les images d'apprentissage. On classe donc 120-20, 100 images. Après exécution, on obtient un taux de 67%. Comparées aux exigences (taux de 100%), le taux est bas certes, mais il ne faut pas oublier que nous utilisons la méthode du proche voisin donc cela paraît normal, pour augmenter ce taux on pourrait étendre aux 3 plus proches voisins.

taux	67

Question n°18 :

- Analyser et comparer les taux obtenus lorsque la distance de Manhattan est remplacée par la distance euclidienne.

```
function [classe]=classifieur(attributs_normalises,img_gris,moyenne,ecart_type,num_classe)
    [aire,perimetre,longueur,largeur]=AttributsForme(img_gris);
    res=[aire,perimetre,longueur,largeur];
    for i=1:4
        res(i)=(res(i)-moyenne(i))./ecart_type(i);
    end
    for app = 1:20
        //tableau_traitement=sqrt(abs(attributs_normalises(:,i)-res(i))^2);
        //distance(app)=sum(abs(attributs_normalises(app,:)-res(1,:)));
        distance(app)=sqrt(sum((attributs_normalises(app,:)-res(1,:)).^2));
    end
    [dist_mini,ind] = min(distance);
    classe=num_classe(ind);
endfunction
```

On ré-exécute le code de la question précédente avec la fonction classifieur modifiée et on trouve un taux de 65% donc inférieur au taux obtenu avec la distance de Manhattan.

taux	65

Question n°19 :

- ❖ Même question lorsque le classifieur du plus proche voisin est remplacé par le classifieur du plus proche barycentre.

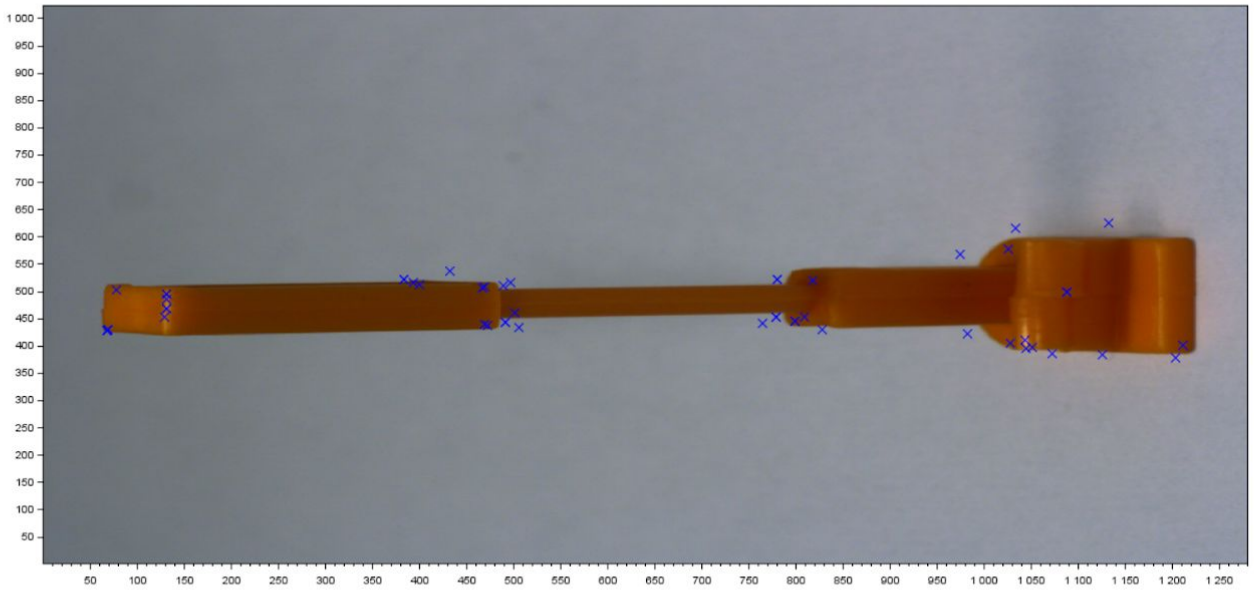
```
function [classe]=classifieur_barycentre(attributs_normalises,img_gris,moyenne,ecart_type,num_classe)
    C=zeros(1,4);
    [aire,perimetre,longueur,largeur]=AttributsForme(img_gris);
    res=[aire,perimetre,longueur,largeur];
    for i=1:4
        res(i)=(res(i)-moyenne(i))./ecart_type(i);
    end

    l=1;
    for i=1:4
        attributs_barycentre(i,:)=[sum(attributs_normalises((l:l+4),1))/5,sum(attributs_normalises((l:l+4),2))/5,sum(attributs_normalis
es((l:l+4),3))/5,sum(attributs_normalises((l:l+4),4))/5];
        l=l+5;
    end

    for app = 1:4
        distance(app)=sum(abs(attributs_barycentre(app,:)-res(1,:)));
    end

    [dist_mini,ind] = min(distance);
    classe=ind;
endfunction
```

Nous avons créé une nouvelle fonction “ classifieur_barycentre ” qui va classer l’image entrante en utilisant la méthode du plus proche barycentre. Pour créer les barycentres de chaque classe on a moyenné leurs attributs. Avec cette méthode, on obtient un taux encore plus bas, un taux de 64%.

Question n°20 :

- A l’aide des fonctions “ *imdetect_SIFT* ” et “ *imextract_DescriptorSIFT* ”, extraire les descripteurs SIFT de chacune des images couleur de la base et procéder à la classification.

Tout d’abord un petit aperçu de ce que sont les éléments SIFT :



Ensuite, effectuons l’apprentissage :

```
nb_classe = 4;
nb_image =30;
nb_ima = nb_classe*nb_image;
nb_ima_train=nb_ima/6;
ima_label=0;
max_ligne=1972;
for i_train=1:nb_ima
    if(((i_train>=1)&&(i_train<=5))||((i_train>=31)&&(i_train<=35))||((i_train>=61)&&(i_train<=65))||((i_train>=91)&&(i_train<=95))) then
        ima_label=ima_label+1;
        num_classe(ima_label)=floor((i_train-1)/nb_image)+1;
        if(i_train/10 <1)
            fichier_train = strcat(['Base\00000',msprintf('%d',i_train),'.png']);
        else
            if(i_train/100<1)
                fichier_train = strcat(['Base\0000',msprintf('%d',i_train),'.png']);
            else
                fichier_train = strcat(['Base\000',msprintf('%d',i_train),'.png']);
            end
        end
        Ima_train=imread(fichier_train);
        fobj=imdetect_SIFT(Ima_train);
        des=imextract_DescriptorSIFT(Ima_train,fobj);
        apprentissage_temp(:,:,ima_label)=[des;    zeros(max_ligne-length(des(:,1)),128)];
        //normalisation(apprentissage_temp,ima_label);

    end
end
```

Pour trouver cette valeur, j'ai du ouvrir toutes les images et extraire les descripteurs SIFT de chacune d'elles. Certaines images ont beaucoup de points caractéristiques d'autres non, ce qui fait que les matrices <<des>> n'ont pas toutes le même nombre de lignes. Cependant le nombre de colonnes reste le même : 128

Ici on construit une matrice de matrice mais plus important encore, on formate les matrices <<des>> pour qu'elles aient toutes 1972 lignes, on met des 0 pour remplir.

Une fois la base d'apprentissage créée nous avons entamé la classification, mais avant faisons un point sur l'outil utilisé afin de classifier les images. Étant donné que l'on compare des matrices, nous avons utilisé la corrélation.

```
function [mu]=calc_mu(des)
    [X,Y]=size(des);
    mu=sum(des)/(X*Y);
endfunction
```

```
function [sigma] = calc_sigma(des)
    mu=calc_mu(des);
    sigma=sqrt(sum((des-mu).^2));
endfunction
```

```
exec calc_mu.sce
exec calc_sigma.sce


function [correlation]=calc_correlation(des_entrante,des_apprentissage)
    mu_apprentissage = calc_mu(des_apprentissage);
    mu_entrante =calc_mu(des_entrante);
    sigma_apprentissage = calc_sigma(des_apprentissage);
    sigma_entrante = calc_sigma(des_entrante);

    correlation = sum((des_entrante-mu_entrante).*(des_apprentissage-mu_apprentissage))/(sigma_apprentissage*sigma_entrante);
endfunction
```

C'est une application concrète de la formule du cours. On prendra la valeur de corrélation la plus élevée pour attribuer une classe à l'image. Cette étape achevée, on passe enfin à la classification :

```
exec classifieur_sift.sce

ima_label=0;
reussite=0;
for i_train=1:nb_ima
    if(((i_train>=6)&&(i_train<=30))||((i_train>=36)&&(i_train<=60))||((i_train>=66)&&(i_train<=90))||((i_train>=96)&&(i_train<=120)))
        ima_label=ima_label+1;
        num_classe_test(ima_label)=floor((i_train-1)/nb_image)+1;
        if(i_train/10 <1)
            fichier_train = strcat(['Base\00000',msprintf('%d',i_train),'.png']);
        else
            if(i_train/100<1)
                fichier_train = strcat(['Base\0000',msprintf('%d',i_train),'.png']);
            else
                fichier_train = strcat(['Base\000',msprintf('%d',i_train),'.png']);
            end
        end
        Ima_train=imread(fichier_train);

        classe_estimee=classifieur_sift(apprentissage_temp,Ima_train);

        if(classe_estimee==num_classe_test(ima_label))
            reussite=reussite+1;
        end

    end
end

taux=(reussite*100)/100;
```

Grâce à cette méthode on obtient un taux de 74%, qui dépasse tous les taux calculés précédemment.

taux	74

Question n°21 :

- Étudier l'influence de l'espace couleur utilisé sur le taux de classification obtenu grâce à la fonction " *rgb2hsv* ".

Tout d'abord il a fallu modifier le nombre de lignes maximum, il était de 1972 pour les images en rgb, ici en hsv il a fallu l'augmenter à 4062. Ensuite nous avons lancé la classification et nous avons trouvé un taux de 54% donc le taux a drastiquement baissé en changeant l'espace de couleurs des images.

taux	54

Question n°22 :

- Comment pourrait-on améliorer les résultats de classification obtenus avec le descripteur SIFT?

On pourrait commencer par essayer de classifier en hsl ou en nuances de gris. On pourrait augmenter le nombre de features dans la fonction "imdetect_SIFT", jouer sur le nombre d'octaves ainsi que la valeur des seuils pour les contours et les faibles textures.

CONCLUSION

A travers ce TP nous avons pu concevoir un algorithme de reconnaissance d'image. Le début du sujet nous a permis de traiter les images afin de les rendre plus faciles à étudier pour notre algorithme (affinage de l'image pour réduire le nombre de régions, détermination du cadre circonscrit, trouver le périmètre…).
Ensuite dans la deuxième partie de ce TP nous avons mis au point notre algorithme permettant de déterminer la classe d'une image (panneaux, bonhommes, voitures ou camions). Le script d'apprentissage nous ayant été fourni, il ne nous restait plus qu'à trouver la meilleure manière de classer ces images (méthode du plus proche voisin/barycentre).

Ce TP nous a permis de mettre en place la création d'un projet réunissant les connaissances que nous avons acquises en cours de Traitement d'Images et Vision, tout en gérant notre organisation pour réussir à terminer cet algorithme de grande taille.