

TD AVIO

31 août 2020

Moments

I est une matrice de dimension (n,m) dont les coefficients $(I_{x,y})_{1 \leq x \leq n, 1 \leq y \leq m} \in \{0, 1\}$ représentant une image binarisée.

Moment cartésien

$$m_{p,q} = \sum_{x=1}^n \sum_{y=1}^m I(x, y) \cdot x^p \cdot y^q$$

Pour une implémentation sur Matlab le plus simple est de voir cette opération comme un produit matriciel (Opération natif)

$$m_{p,q} = Y \cdot I \cdot X^T \text{ avec } X = [1, 2, \dots, n]^p \text{ et } Y = [1, 2, \dots, m]^q$$

Moment centraux

$$\mu_{p,q} = \sum_{x=1}^n \sum_{y=1}^m I(x, y) \cdot (x - \mu_x)^p \cdot (y - \mu_y)^q$$

Avec $(\mu_x, \mu_y) = \left(\frac{m_{1,0}}{m_{0,0}}, \frac{m_{0,1}}{m_{0,0}} \right)$ Le barycentre de l'objet

$$\mu_{p,q} = X \cdot I \cdot Y^T \text{ avec } X = ([1, 2, \dots, n] - \mu_x)^p \text{ et } Y = ([1, 2, \dots, m] - \mu_y)^q$$

Moment invariant à la translation

Moments centraux normalisés

$$\eta_{p,q} = \frac{\mu_{p,q}}{m_{0,0}^{\frac{p+q}{2} + 1}}$$

Moment invariant à l'échelle

Soit la fonction suivante :

```

1 function [result] = Moment(I, param)
2 [m,n] = size(I);
3 x = 1:n;
4 y = 1:m;
5 norm = 1;
6 while 1 % Simule un switch style Java/C/C++/...
7     switch param(3)
8         case 2
9             norm = Moment(I, [0, 0, 0])^((param(1)+param(2))/2+1);
10            param(3) = 1;
11        case 1
12            x = x - Moment(I, [1, 0, 0])/Moment(I, [0, 0, 0]);
13            y = y - Moment(I, [0, 1, 0])/Moment(I, [0, 0, 0]);
14            break;
15        otherwise
16            break;
17    end
18 end
19 result = (y.^param(2))*I*(x.^param(1))/norm;
20 end

```

Forme	$m_{0,0}$	$\mu_{0,0}$	$\eta_{0,0}$	$m_{1,1}$	$\mu_{1,1}$	$\eta_{1,1}$
♥	3850	3850	1	42678142	136	9.23e-06
♦	3746	3746	1	58334614	-4217	-3e-04
♥	8677	8677	1	104217234	1266	1.68e-05
♦	7339	7339	1	95847046	410	7.62e-06
♣	4179	4179	1	51883826	-87111	-0.004
♣	4166	4166	1	50246390	937	5.40e-05
♣	4148	4148	1	54812470	1119	6.50e-05
♣	9384	9384	1	116010638	-11157	-1.26e-04

Segmentation



Sur l'image il a 4 markers de couleurs différentes (noir, rouge, vert, bleu) on cherche à mettre en relation l'estimation du barycentre des bouchons à sa couleur. À priori on va pouvoir les discriminer grâce à leurs couleurs.

Base de colorimétrie

La représentation des couleurs en base RGB est la plus pertinente et en tendance nous avons :

Couleur	R	G	B	R-G	R-B	G-B	Maximisé
Noir	0	0	0	0	0	0	255-R-G-B
Rouge	1	0	0	1	1	0	2R-G-B
Vert	0	1	0	-1	0	1	2G-B-R
Bleu	0	0	1	0	-1	-1	2B-R-G

J'introduis artificiellement les colonnes différences pour rendre compte de la condition sur les 3 composantes. La colonne "**maximisé**" permet de mesurer à quel point le pixel appartient à la couleur correspondante. La mesure et donc une combinaison des colonnes différences de sorte à avoir une mesure positive. On peut ainsi former une nouvelle image à 4 dimensions avec une base colorimétrique différente basée sur cette mesure.

```
1 function [image, imageBin, color] = BaseMesureColor(I, type, norm)
2     [n,m, t] = size(I);
3     % Composantes RGB
4     r = type(I(:,:,1));
5     g = type(I(:,:,2));
6     b = type(I(:,:,3));
7     % Changement de base, mesure de la couleur
8     image = reshape(norm*[2*r-g-b, 2*g-b-r, 2*b-r-g, 255-r-g-b],[n,m,4]);
9     imageBin(:,:,1) = image(:,:,1) > 40;
10    imageBin(:,:,2) = image(:,:,2) > 10;
11    imageBin(:,:,3) = image(:,:,3) > 30;
12    imageBin(:,:,4) = image(:,:,4) > 112;
13    color = {'red', 'green', 'blue', 'black'};
14 end
```

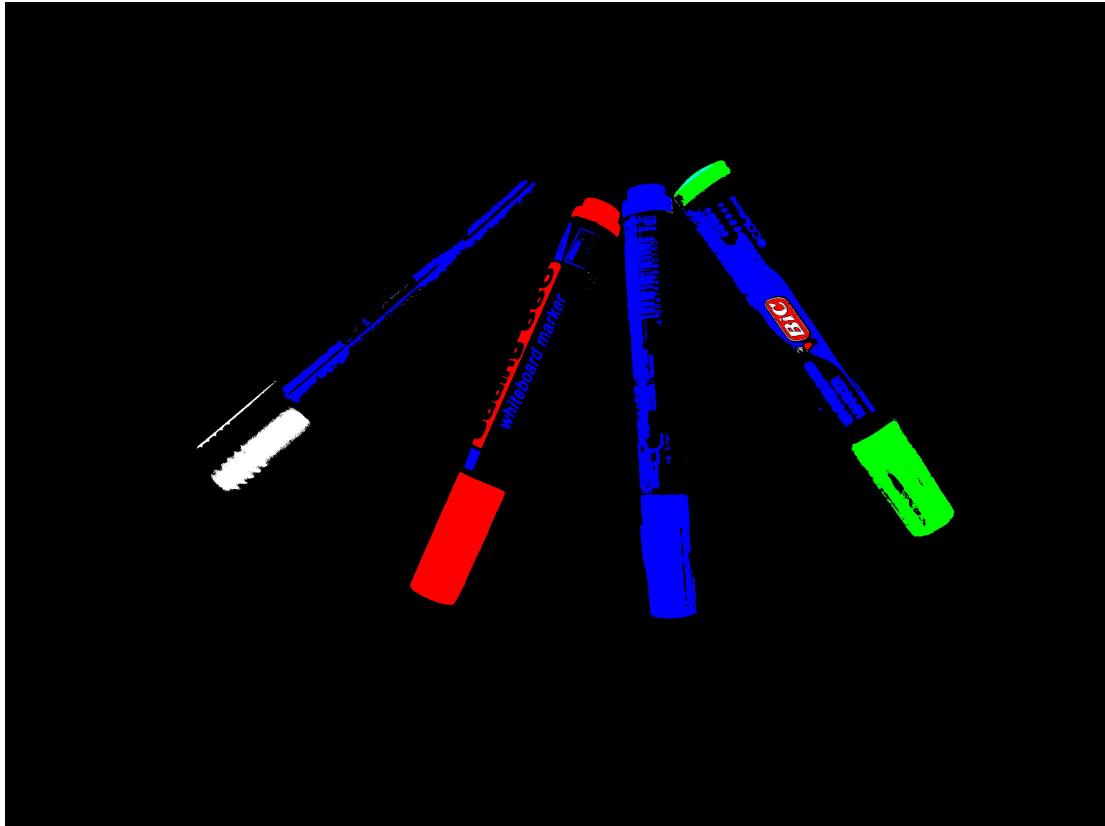


FIGURE 1 – Combinaison des 3 premières composantes (RGB)

Les parties négatives sont mises à zéros.

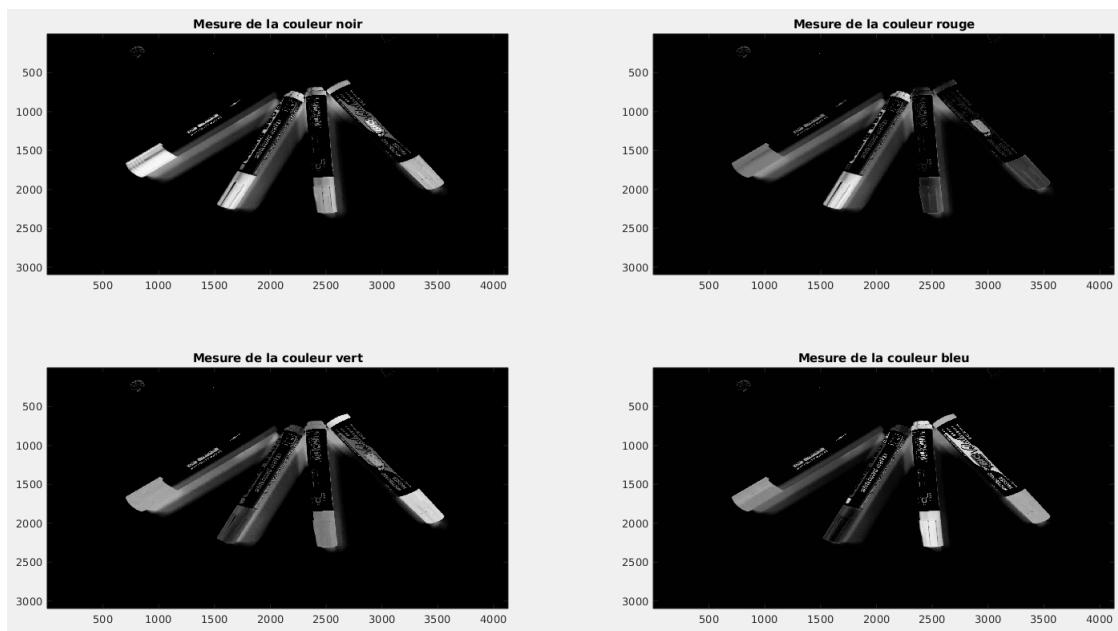


FIGURE 2 – Image en nuances de gris pour des 4 composantes (Black-RGB)

A priori sur l'image un bouchon est l'élément qui représente le mieux sa couleur. On peut appliquer un simple seuillage qui conserve le mieux la forme du bouchon tous en écartant les objets de la même couleur.

Seuillage

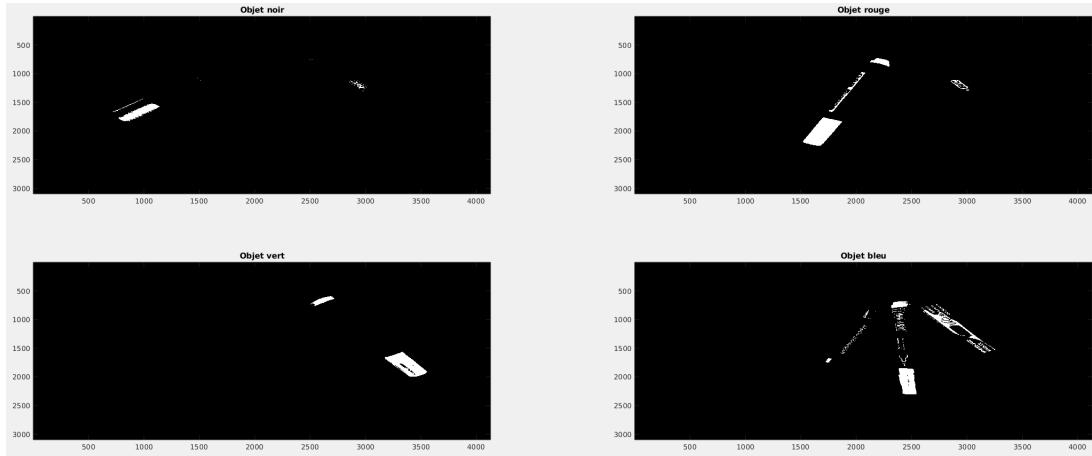


FIGURE 3 – Image en nuances de gris pour des 4 composantes (Black-RGB) après seuillage

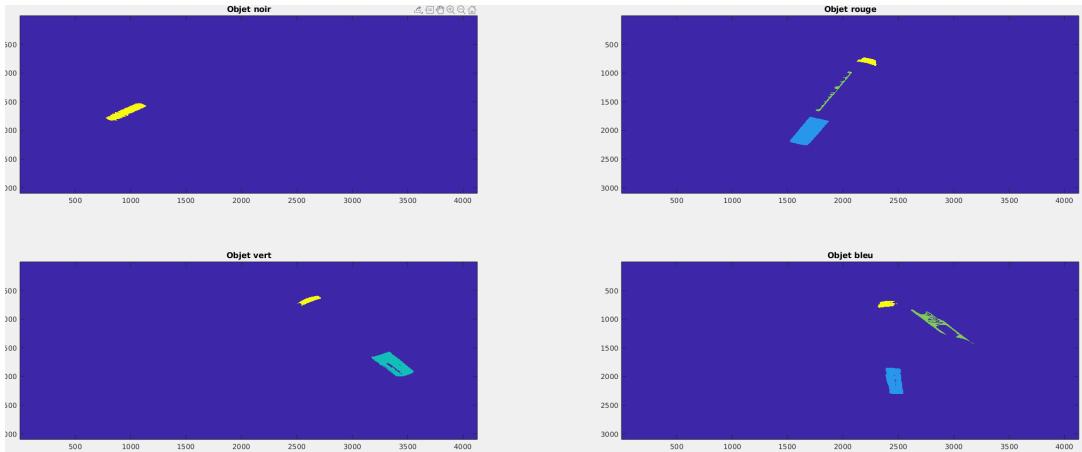
Les bouchons sont plutôt bien segmentés visuellement on les reconnaît assez facilement. L'avantage avec ce changement de base est qu'il rend plus simple le seuillage.

Extraction des zones

```

1 function [ENoArtefact] = Zone(Im, sizemin)
2     i = 1;
3     [n,m] = size(Im);
4     E = zeros(n,m,1);
5     for x = 1:n
6         for y = 1:m
7             % Si le point est un élément d'un objet
8             if(Im(x, y) == 1)
9                 if(E(x,y-1) ~= 0 || E(x-1,y) ~= 0)
10                    % Si une zone est adjacente au point (Gauche, Haut)
11                    % Inutile de tester en (bas, droite) on est pas encore
12                    % passé
13                    if(E(x,y-1) ~= 0 && E(x-1,y) ~= 0)
14                        %Si deux zones sont adjacentes
15                        if(E(x-1,y) ~= E(x,y-1))
16                            % Il est possible qu'on assemble deux zones
17                            % différentes dans ce cas fusionner les deux
18                            % zones par la gauche.
19                            % Trés couteux en temps pas moyen de faire
20                            % fonctionner les pointeurs correctement ***
21                            E(E==E(x-1,y)) = E(x,y-1);
22                        end
23                        E(x,y) = E(x,y-1);
24                    elseif(E(x,y-1) ~= 0)
25                        % Si une zone à gauche
26                        E(x,y) = E(x,y-1);
27                    elseif(E(x-1,y) ~= 0)
28                        % Si une zone en haut
29                        E(x,y) = E(x-1,y);
30                    end
31                else
32                    % Si aucune zone est autour de ce point => création
33                    % d'une nouvelle zone avec un nouvelle index
34                    E(x,y) = i;
35                    i = i +1;
36                end
37            end
38        end
39        disp([num2str(x,double(n)*100), ' %'])
40    end
41    %Calcule la surface des zones et retire celles qui font moins de
42    %sizemin puis les tries dans l'ordre décroissant
43    number = unique(E);
44    surface = zeros(length(number),2);
45    for i = 1:length(number) %Relation entre la zone et la surface
46        surface(i,1) = number(i);
47        surface(i,2) = length(E(E==number(i)));
48    end
49    %Retire les zones dont la surface fait moins que sizemin
50    surfaceNoArtefact = surface(find(surface(:,2) > sizemin & surface(:,1) ~=0 ),:)
51    %Triage sur les surfaces décroissant
52    [values,idx] = sort(surfaceNoArtefact(:,2), 'descend');
53    sortSurfaceNoArtefac = surfaceNoArtefact(idx,:);
54    %Attribue de nouveaux index aux zones
55    ENoArtefact = zeros(n,m);
56    e = 1;
57    for i = sortSurfaceNoArtefac(:,1)'
58        ENoArtefact(E == i) = e;
59        e = e+1;
60    end
61 end

```



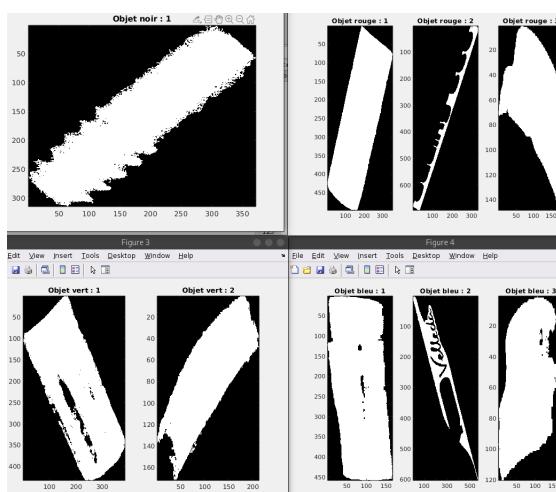
Chaque objet connexe de surface suffisante est indexé.

1 ROI

```

1 function [result] = Roi(im)
2     result = zeros(max(unique(im)), 4);
3     [n,m] = size(im);
4     for value = 1:max(unique(im))
5         %Le premier find et le dernier
6         %correspond à xmin, xmax
7         xIndex = mod(find(im == value),n);
8         %Rotate 90 degré l'image pour que le premier find et le dernier
9         %correspond à ymin, ymax
10        yIndex = mod(find(im' == value),m);
11        result(value, 1) = min(xIndex);
12        result(value, 2) = min(yIndex);
13        result(value, 3) = max(xIndex);
14        result(value, 4) = max(yIndex);
15    end
16 end

```



Les objets sont extraits des images et ordonnés par taille de surface. Pour la première image prendre le premier objet et calculer son barycentre est suffisant.

Résultats

Première image test



On a une bonne estimation des barycentres des bouchons associés à sa couleur conformément au cahier des charges.

Deuxième image test

On nous propose une deuxième image qui ajoute une étoile noire. Elle va poser soucie pour le barycentre du bouchon noir.

On peut imaginer beaucoup de moyens pour discriminer l'étoile d'un bouchon, le plus simple est d'utiliser une mesure de compacité associer à la surface de l'objet pour discriminer ces deux éléments.

L'application reste assez générale si un veut le barycentre d'un bouchon jaune il suffit de rajouter



une composante dans la base colorimétrique maximisant le jaune. Les bouchons sont bien identifiés malgré la présence d'un objet semblable au bouchon noir.