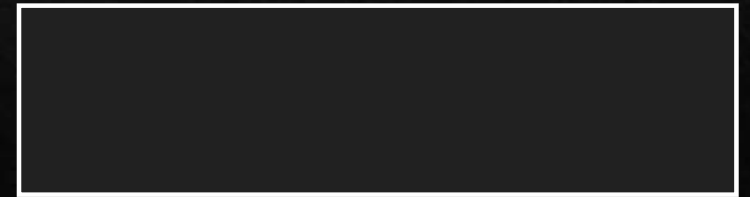




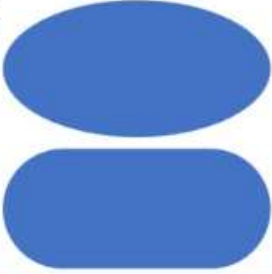

La vision par  
ordinateur au service  
de la détection des  
bactéries.



# Plan

- ◇ Mise en situation et description du problème
- ◇ Méthode mathématique
  - ◇ Introduction.
  - ◇ Etapes de la méthode.
  - ◇ Amélioration
  - ◇ Conclusion
- ◇ Utilisation de OpenCV
  - ◇ Introduction
  - ◇ Conclusion
- ◇ Comparaison des méthodes & conclusion

# Mise en situation et description du problème

NOM DE LA FAMILLE	FORME	EXEMPLES	En réalité
Cocci	Arrondie 	Staphylocoque doré Streptocoque	
Bacilli	Bâtonnet 	Escherichia coli	
...	...	...	...

Partie 1 :

- Tableau à information
- Détermination du contour

Partie 2 :

- Classification morphologique
- Densité surfacique de bactéries

# Mise en situation et description du problème



Details [Mode Urine] Version 00-24 Build014 of 2022/05/25 15:05

Menu List résultat Détails Précédent Suivant 10 Derniers Rechercher Valider Effacer

Validated No 502205250982  
URI 2022/05/25 14:23:24 000008-01

Analyse Research1 Research2

Paramètres	Résultat	Unité	Paramètres	Résultat	Unité
RBC	38.6	/uL	CAST	0.15	/uL
NL RBC	29.6	/uL	Hy.CAST	0.15	/uL
WBC	92.9	/uL	Path.CAST	0.00	/uL
WBC Clumps	0-3	/uL	BACT	1732.9	/uL
EC	31.9	/uL	X'TAL	0.0	/uL
Squa.EC	26.0	/uL	YLC	2.5	/uL
Non SEC	5.8	/uL	SPERM	0.0	/uL
Tran.EC	0.2	/uL	MUCUS	1.0-	/uL
RTEC	5.6	/uL			

Review Comment

Research Information

RBC : Dysmorphic?  
UTI : UTI?  
BACT : Gram Positi...

Start up 3 minute(s) rest



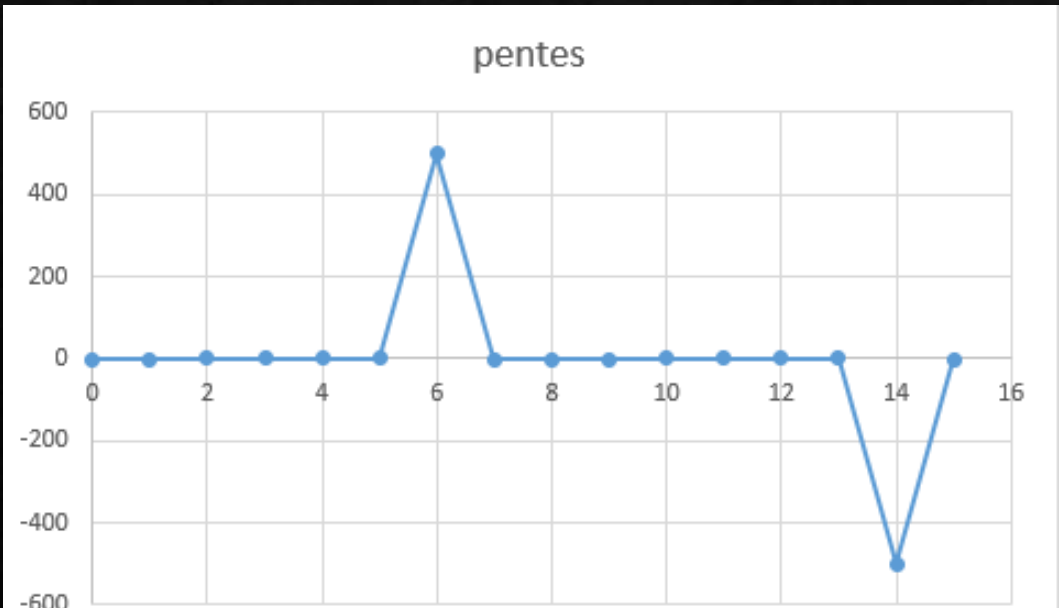
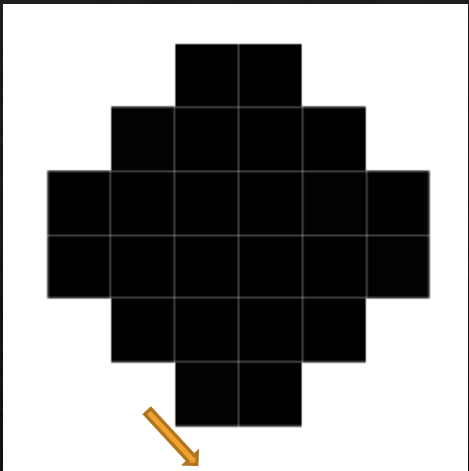
# Plan

- ◇ Mise en situation et description du problème
- ◇ Méthode mathématique
  - ◇ Introduction.
  - ◇ Etapes de la méthode.
  - ◇ Amélioration
  - ◇ Conclusion
- ◇ Utilisation de OpenCV
  - ◇ Introduction
  - ◇ Conclusion
- ◇ Comparaison des méthodes & conclusion

# Méthode mathématique : Introduction

## Démarche Mathématique :

- Déplacement par un pixel dans les 8 directions.
  - 5 pentes possible.
  - Calcule de pentes.
- Comparaison des valeurs à celles des images dans la base de données.



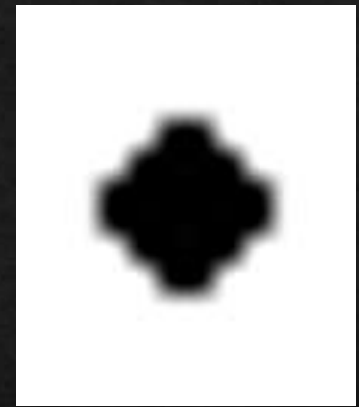
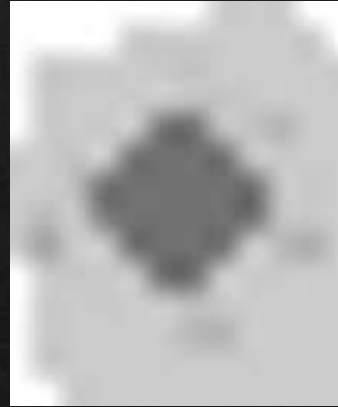
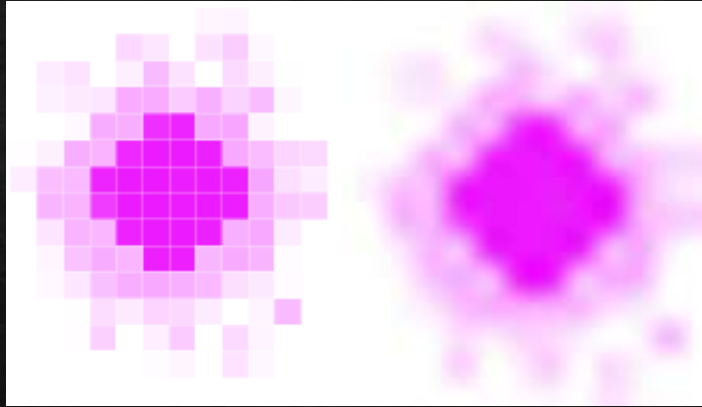
$$a = \frac{y_i - y_{i+1}}{x_i - x_{i+1}}$$

	A	B
1	nombre du pixel	pentes
2	0	-1
3	1	-1
4	2	0
5	3	1
6	4	1
7	5	1
8	6	500
9	7	-1
10	8	-1
11	9	-1
12	10	0
13	11	1
14	12	1
15	13	1
16	14	-500
17	15	-1

# Plan

- ◇ Mise en situation et description du problème
- ◇ Méthode mathématique
  - ◇ Introduction.
  - ◇ Etapes de la méthode.
  - ◇ Amélioration
  - ◇ Conclusion
- ◇ Utilisation de OpenCV
  - ◇ Introduction
  - ◇ Conclusion
- ◇ Comparaison des méthodes & conclusion

# Méthode mathématique : Etapes



$$M_{14,12}(\mathbb{R}^4)$$

Chaque élément de la  
matrice :

[T, B, G ,R]

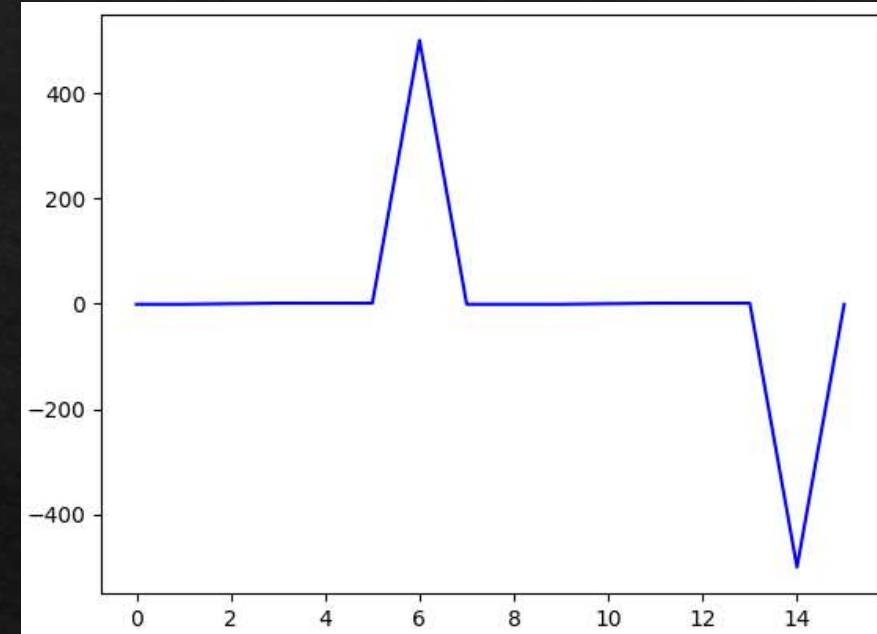
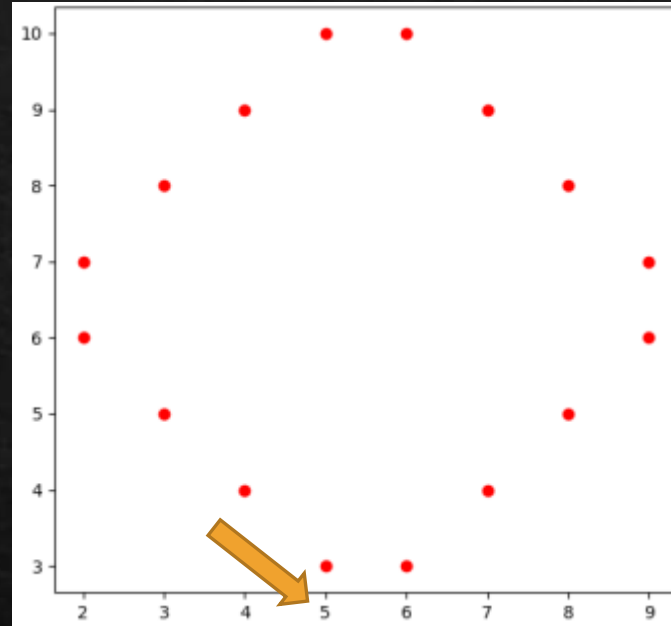
**T = transparence**

```
[255 255 255 255 255 255 255 204 204 204 255 255]
[255 255 255 255 204 204 204 204 204 204 204 255]
[255 204 204 204 204 199 194 204 204 204 204 204]
[255 204 204 204 202 203 194 196 200 204 204 204]
[255 204 204 200 199 119 116 193 193 183 204 204]
[204 204 194 202 119 114 114 116 195 201 204 204]
[204 204 189 117 114 114 114 114 114 197 199 204]
[204 186 201 130 114 114 114 114 114 194 199 204]
[204 166 201 199 115 114 114 115 201 185 175 204]
[255 204 199 201 196 114 114 200 202 202 204 204]
[255 204 204 201 203 202 189 200 204 204 204 204]
[255 204 204 204 204 201 184 182 204 204 204 204]
[255 204 204 204 204 204 204 204 204 204 204 204]
[255 255 204 204 204 204 204 204 204 204 204 204]
```

```
[
  [255 255 255 255 255 255 255 255 255 255 255 255],
  [255 255 255 255 255 255 255 255 255 255 255 255],
  [255 255 255 255 255 255 255 255 255 255 255 255],
  [255 255 255 255 255 255 255 255 255 255 255 255],
  [255 255 255 255 255 0 0 255 255 255 255 255],
  [255 255 255 255 0 0 0 0 255 255 255 255],
  [255 255 255 0 0 0 0 0 0 255 255 255],
  [255 255 255 0 0 0 0 0 0 255 255 255],
  [255 255 255 255 0 0 0 0 255 255 255 255],
  [255 255 255 255 255 0 0 255 255 255 255 255],
  [255 255 255 255 255 255 255 255 255 255 255 255],
  [255 255 255 255 255 255 255 255 255 255 255 255],
  [255 255 255 255 255 255 255 255 255 255 255 255],
  [255 255 255 255 255 255 255 255 255 255 255 255],
  [255 255 255 255 255 255 255 255 255 255 255 255]]
]
```



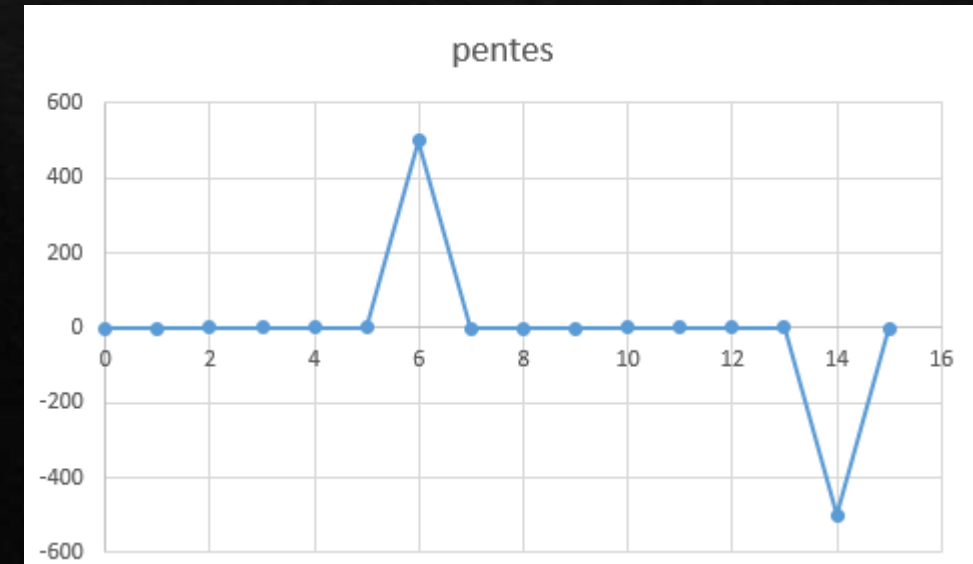
# Méthode mathématique : Etapes



[ -1. -1. 0. 1. 1. 1. 500. -1. -1. -1. -0. 1. 1. 1. -500. -1.]

16 points de contour dans les deux images

$$\frac{1}{N} \sum_{k=0}^N |f_1(\mathbf{x}_k) - f_2(\mathbf{x}_k)| = 0$$

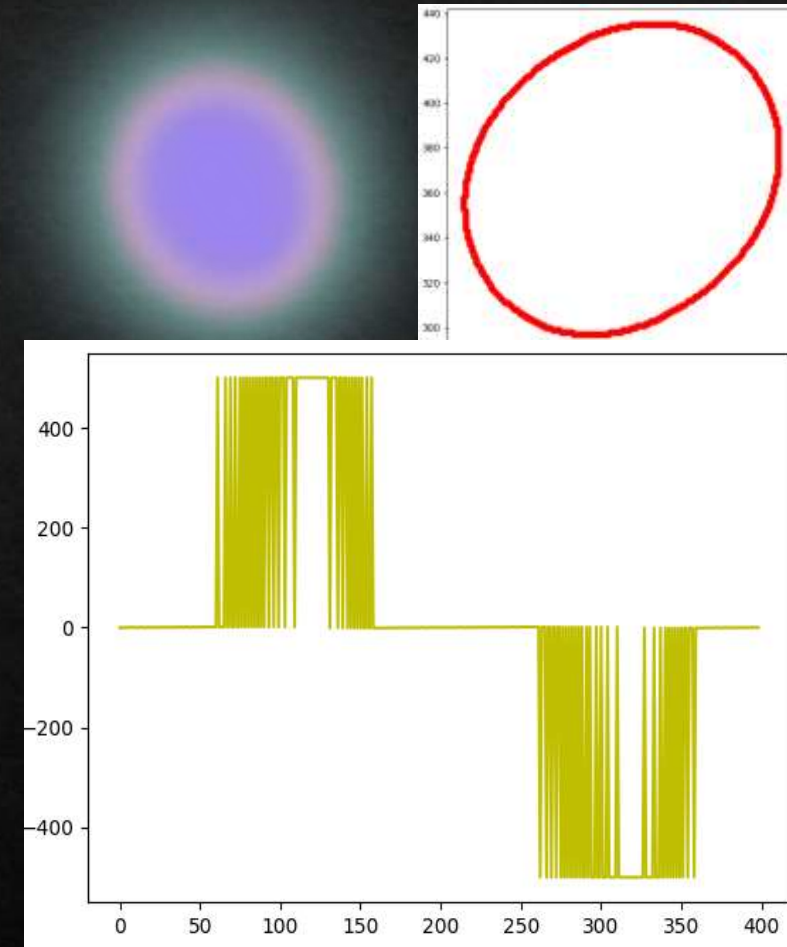
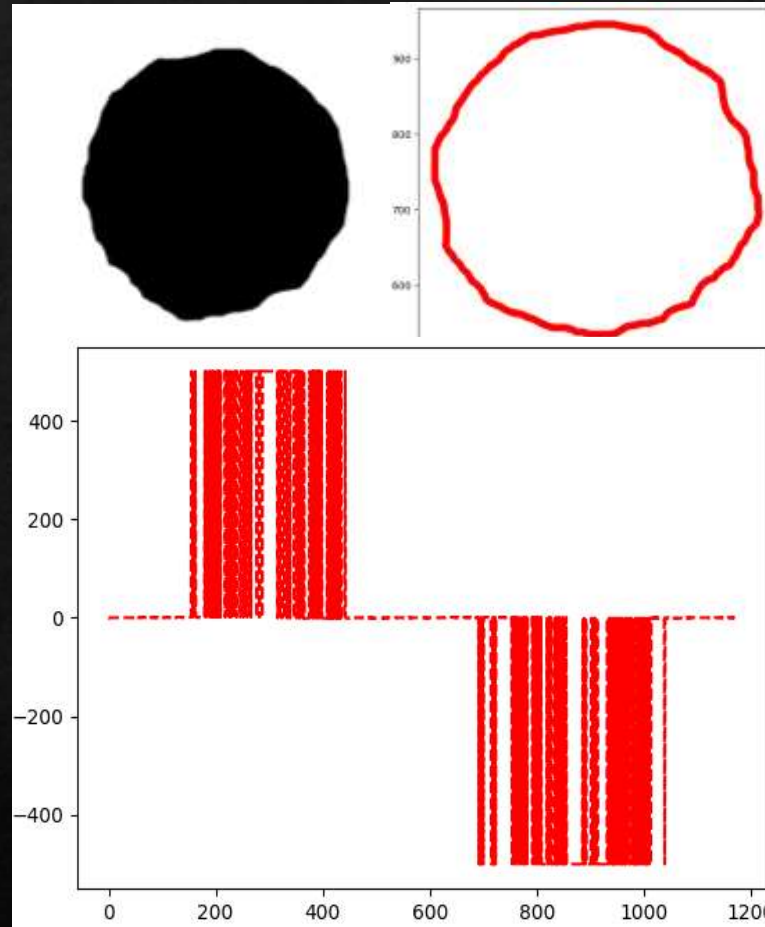
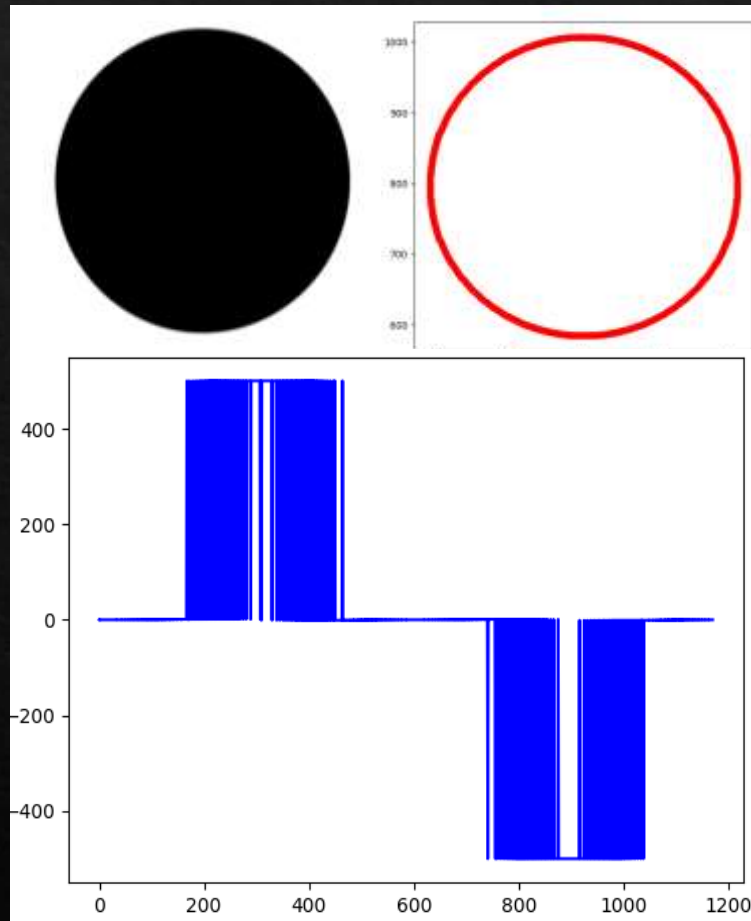


# Plan

- ◇ Mise en situation et description du problème
- ◇ Méthode mathématique
  - ◇ Introduction.
  - ◇ Etapes de la méthode.
  - ◇ Amélioration
  - ◇ Conclusion
- ◇ Utilisation de OpenCV
  - ◇ Introduction
  - ◇ Conclusion
- ◇ Comparaison des méthodes & conclusion

# Méthode mathématique : Amélioration

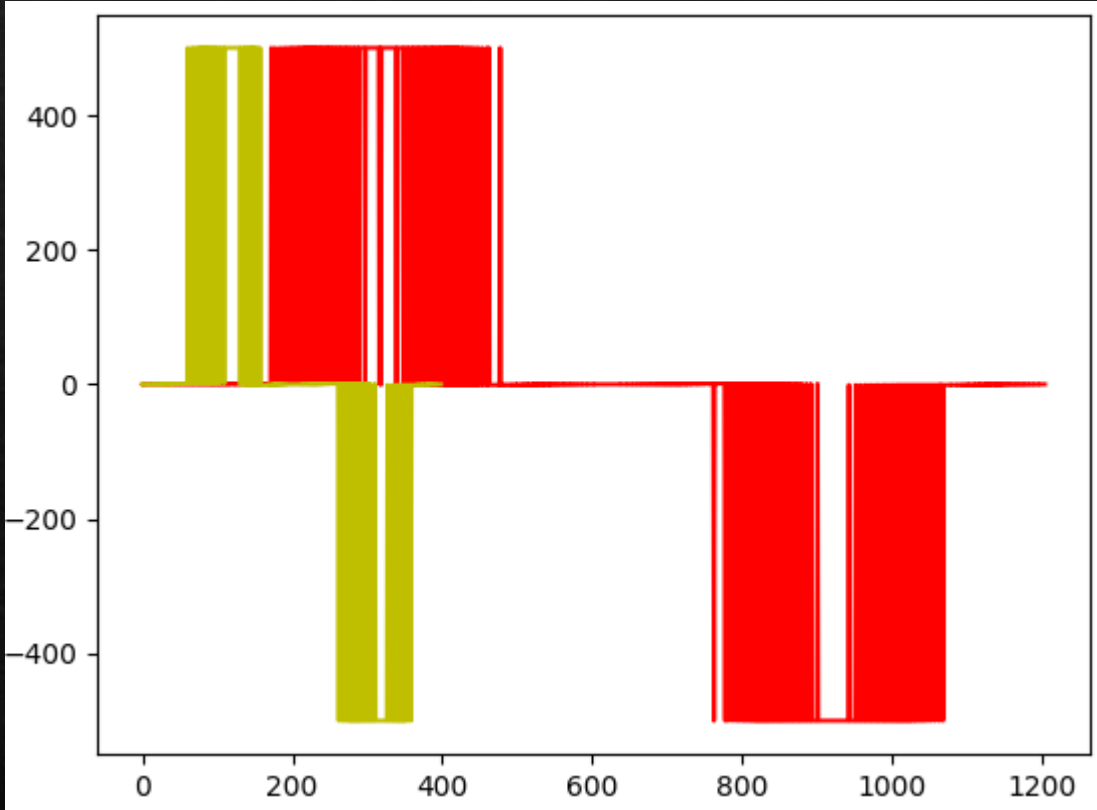
Comparaison (Pour valeur\_seuillage = 160)



$$a = \frac{y_i + c - y_{i+1} - c}{x_i + d - x_{i+1} - d} = \frac{y_i - y_{i+1}}{x_i - x_{i+1}}$$

$$\frac{1}{N} \sum_{k=0}^N |f_1(\mathbf{x}_k) - f_2(\mathbf{x}_k)|$$

# Méthode mathématique : Amélioration

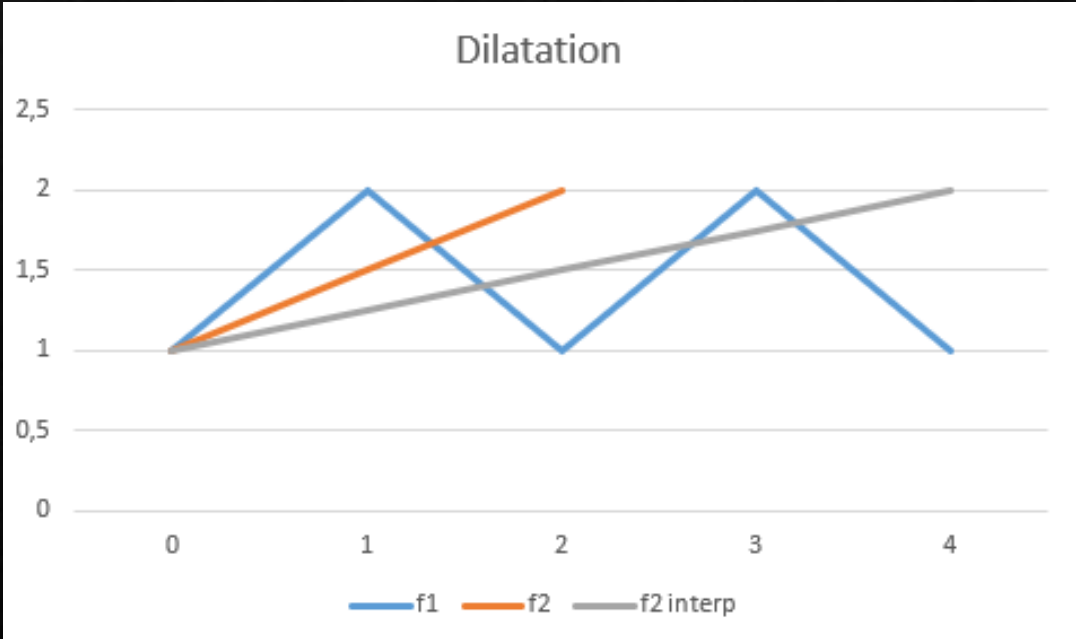


$$f : [0, X] \rightarrow [0, Y]$$
$$x \mapsto ax$$

$$f(X) = Y = aX$$

$$a = \frac{Y}{X}$$

13	abssices	f1	f2	f2 interp
14	0	1	1	1
15	1	2	1,5	1,25
16	2	1	2	1,5
17	3	2		1,75
18	4	1		2





# Méthode mathématique : Amélioration

Interpolation linéaire :

$$f(x) = ax + b$$

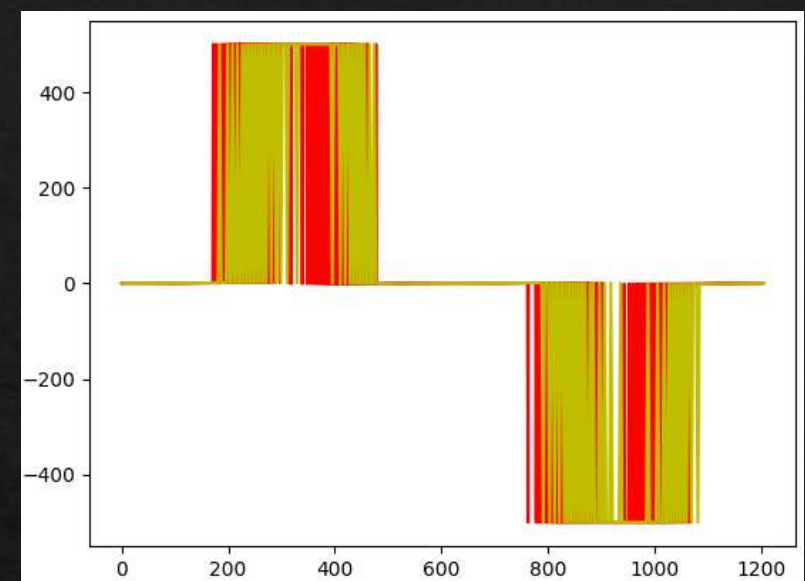
Avec :  $ax_1 + b = y_1$

Et  $ax_2 + b = y_2$

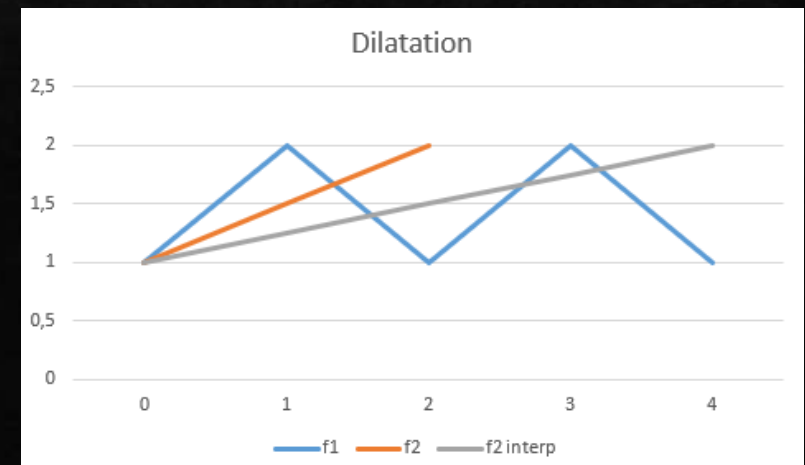
$$\begin{cases} ax_1 + b = y_1 \\ ax_2 + b = y_2 \end{cases}$$

$$a = \frac{y_2 - y_1}{x_2 - x_1} \quad x_2 \neq x_1$$

$$b = \frac{y_2 + y_1 - \frac{(y_2 - y_1)(x_2 + x_1)}{x_2 - x_1}}{2}$$

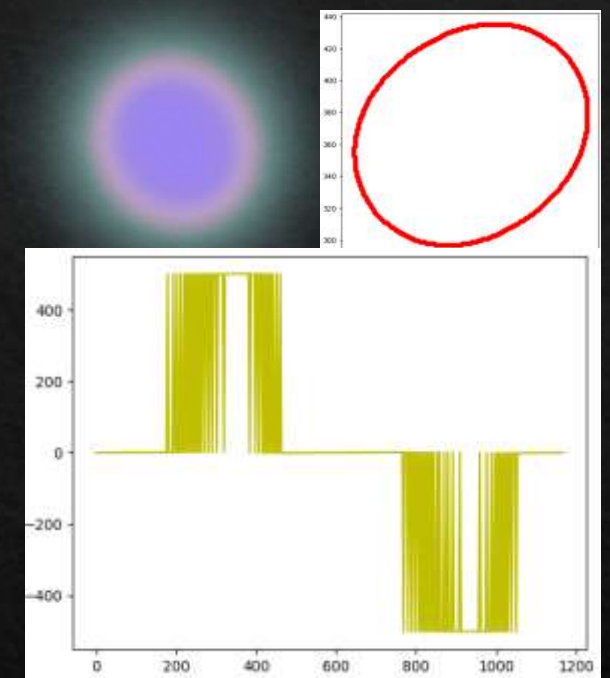
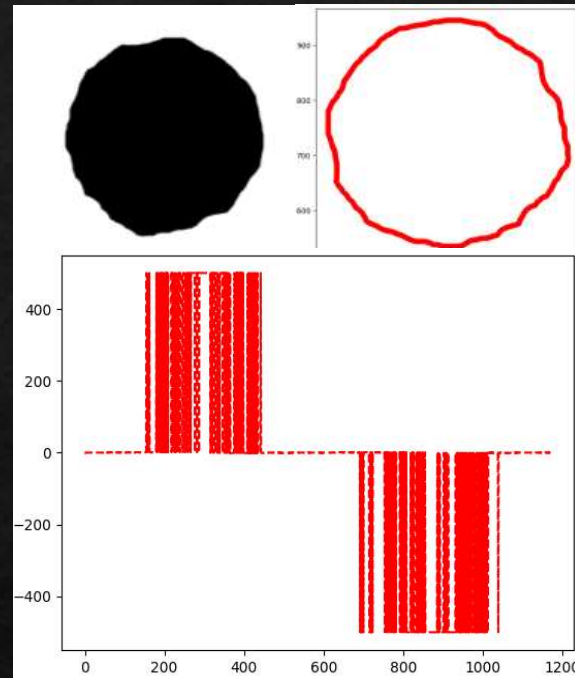
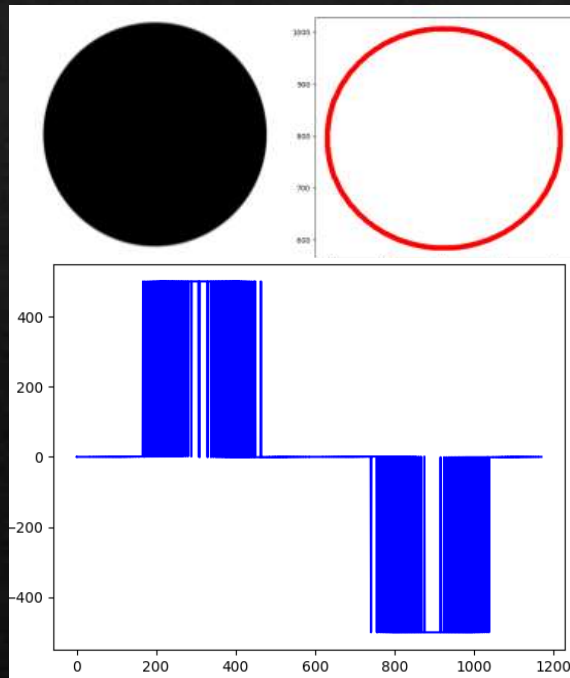


13	abssices	f1	f2	f2 interp
14	0	1	1	1
15	1	2	1,5	1,25
16	2	1	2	1,5
17	3	2		1,75
18	4	1		2



# Méthode mathématique : Amélioration

Comparaison ( Pour valeur\_seuillage = 160 )

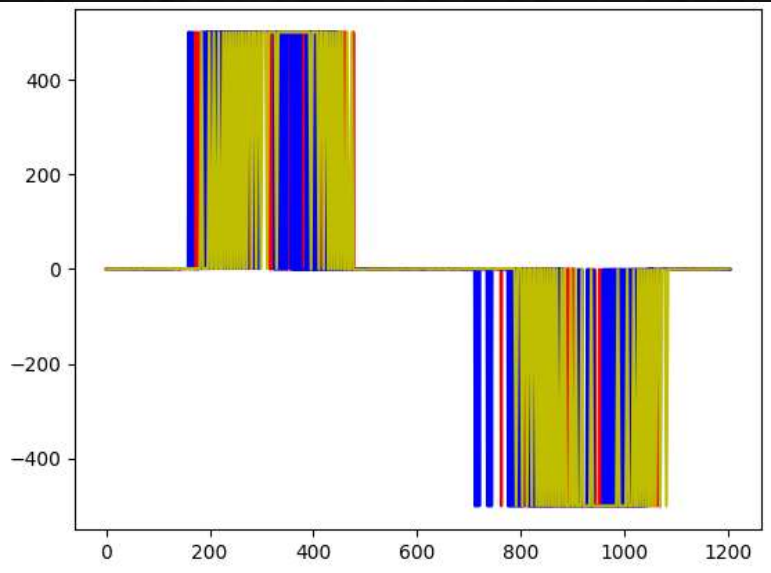
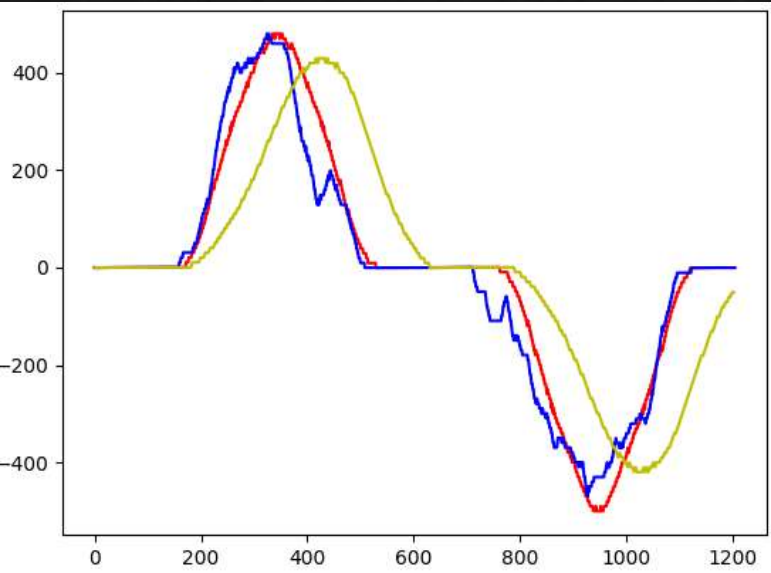


$$\frac{1}{N} \sum_{k=0}^N |f_1(x_k) - f_2(x_k)|$$

G	H
Nom de la photo	Moyenne Abs
rond.png	0
rond_deformé.png	177.81170619961722
ovale.png	156.93872737867406

# Méthode mathématique : Amélioration

n = 50



Moyenne Glissante :

$$\begin{aligned} i < n - 1 & \quad v_i = \sum_{k=0}^i \frac{u_k}{i+1} \\ i \geq n - 1 & \quad v_i = \sum_{k=i-n+1}^i \frac{u_k}{n} \end{aligned}$$

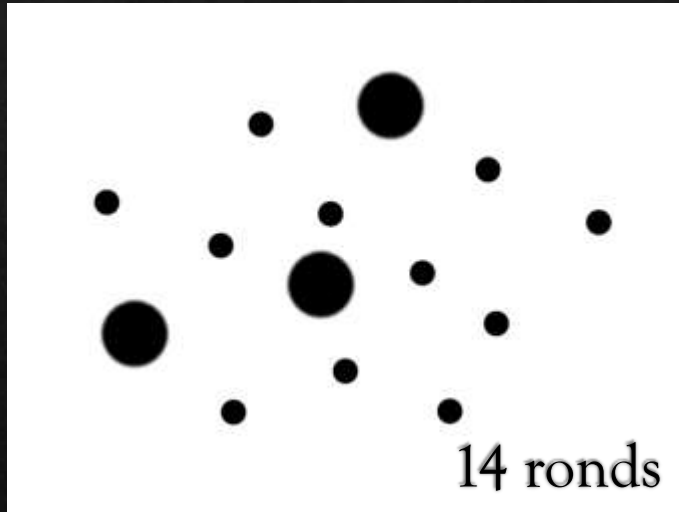
	A	B	C
1	Nom de la photo	MoyenneGlissante	Moyenne Abs
2	rond.png	sans	0
3	rond_deformé.png	sans	177.81170619961722
4	ovale.png	sans	156.93872737867406
5	rond.png	avec (=50)	0
6	rond_deformé.png	avec (=50)	33.01154279266081
7	ovale.png	avec (=50)	110.99051870135388

# Plan

- ◇ Mise en situation et description du problème
- ◇ Méthode mathématique
  - ◇ Introduction.
  - ◇ Etapes de la méthode.
  - ◇ Amélioration
  - ◇ Conclusion
- ◇ Utilisation de OpenCV
  - ◇ Introduction
  - ◇ Conclusion
- ◇ Comparaison des méthodes & conclusion



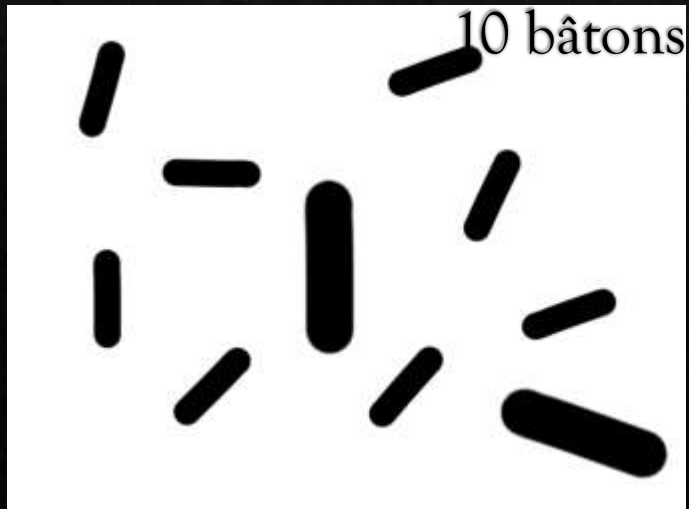
## Méthode mathématique : Conclusion



Le nombre de particules trouvé est : 14

Le nombre de cocci dans l'image donnée est : 14

Le nombre de bacilles dans l'image donnée est : 0



Le nombre de particules trouvé est : 10

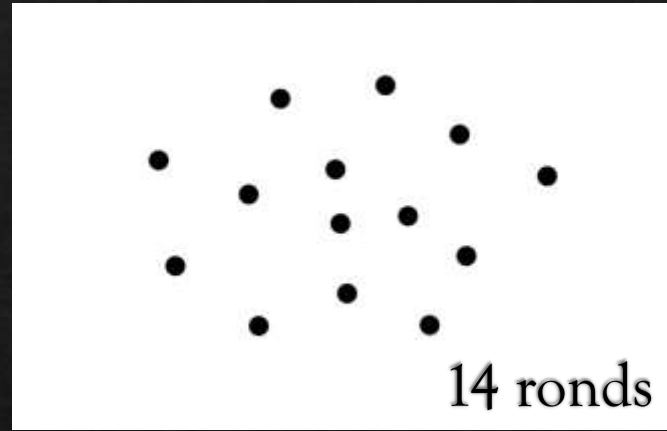
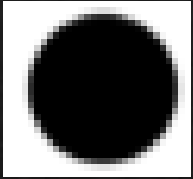
Le nombre de cocci dans l'image donnée est : 2

Le nombre de bacilles dans l'image donnée est : 5

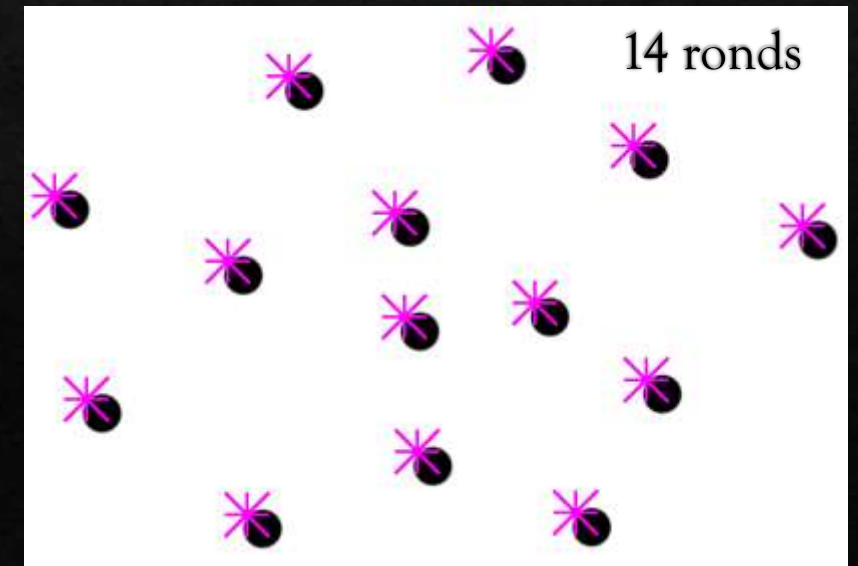
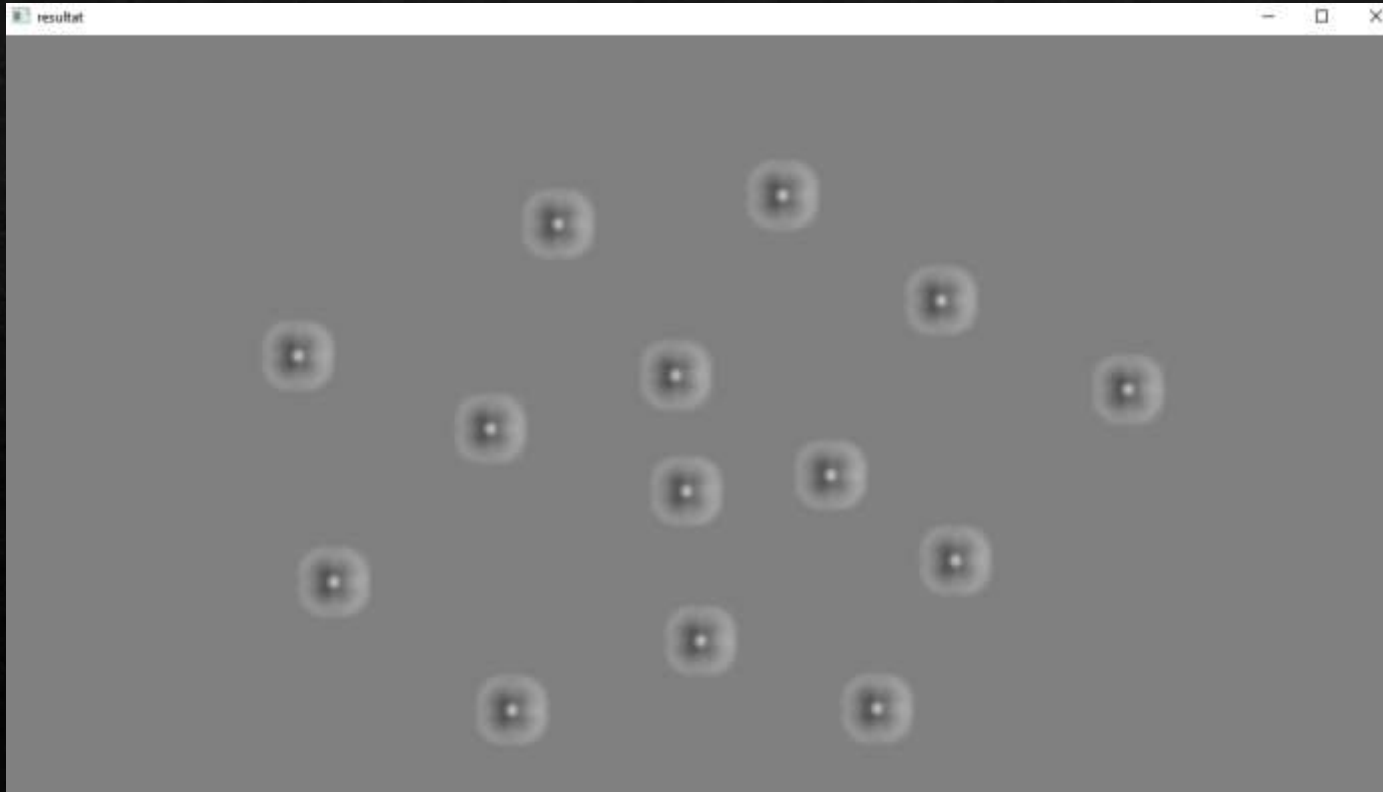
# Plan

- ◇ Mise en situation et description du problème
- ◇ Méthode mathématique
  - ◇ Introduction.
  - ◇ Etapes de la méthode.
  - ◇ Amélioration
  - ◇ Conclusion
- ◇ Utilisation de OpenCV
  - ◇ Introduction
  - ◇ Conclusion
- ◇ Comparaison des méthodes & conclusion

# OpenCV : Introduction



```
Cv2.matchTemplate()  
cv2.groupRectangles()  
Cv2.drawMarker()
```



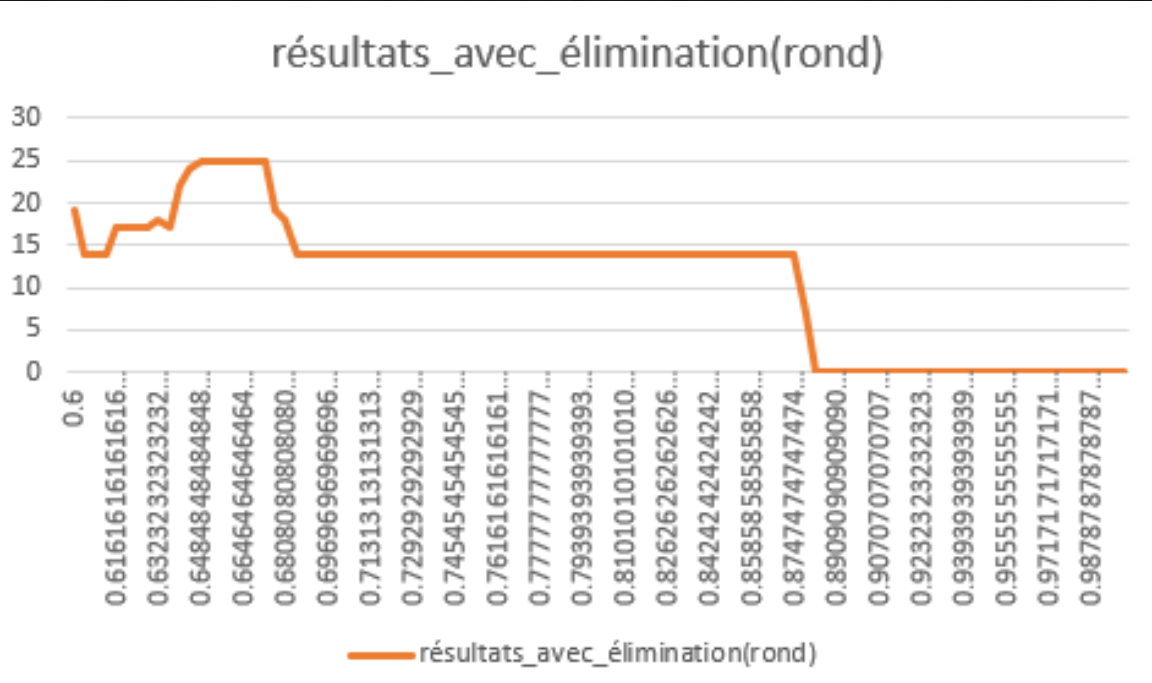
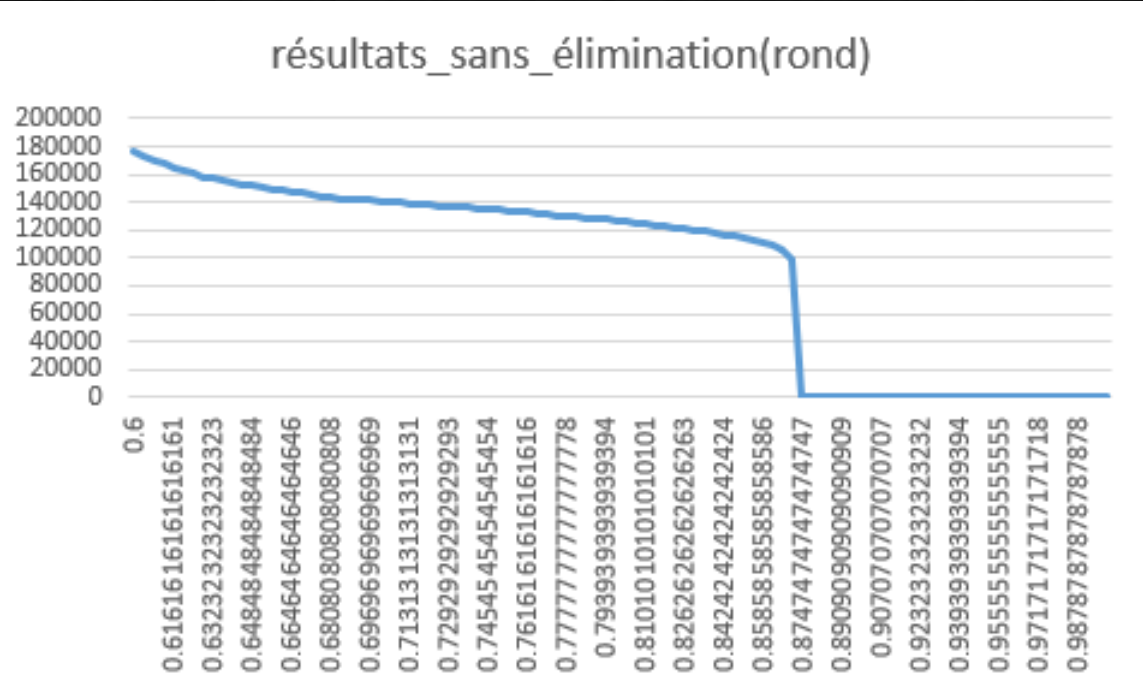
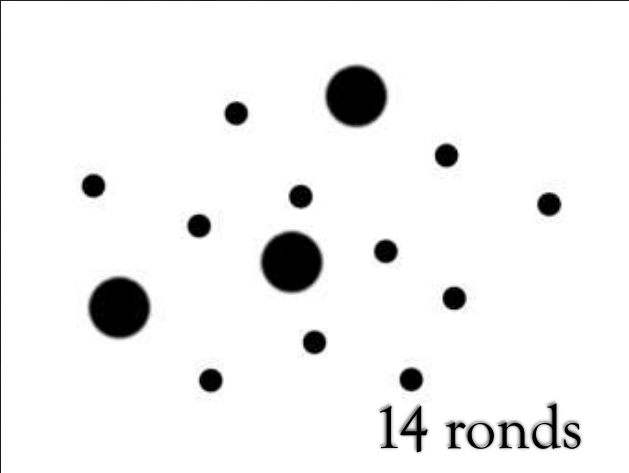
# Plan

- ◇ Mise en situation et description du problème
- ◇ Méthode mathématique
  - ◇ Introduction.
  - ◇ Etapes de la méthode.
  - ◇ Amélioration
  - ◇ Conclusion
- ◇ Utilisation de OpenCV
  - ◇ Introduction
  - ◇ Conclusion
- ◇ Comparaison des méthodes & conclusion



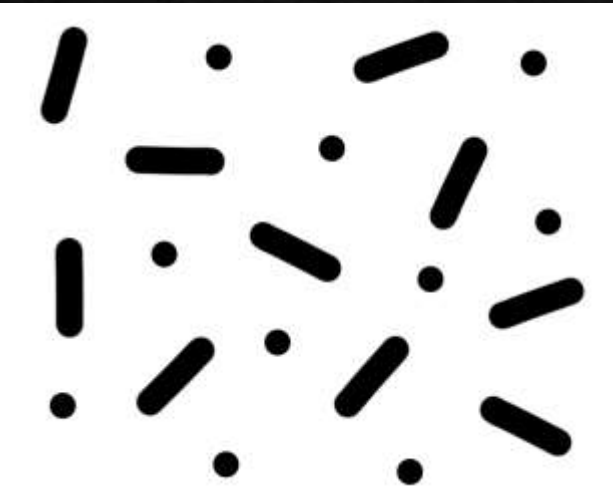
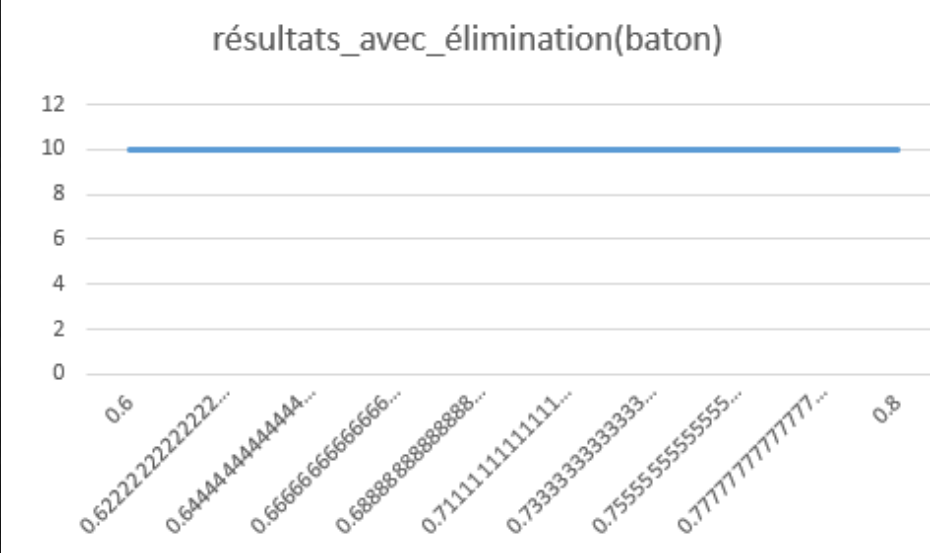
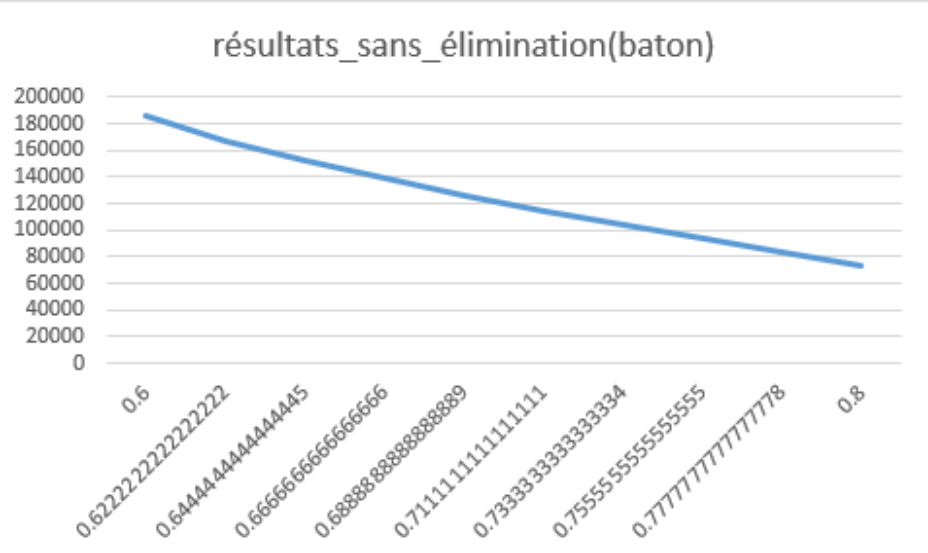
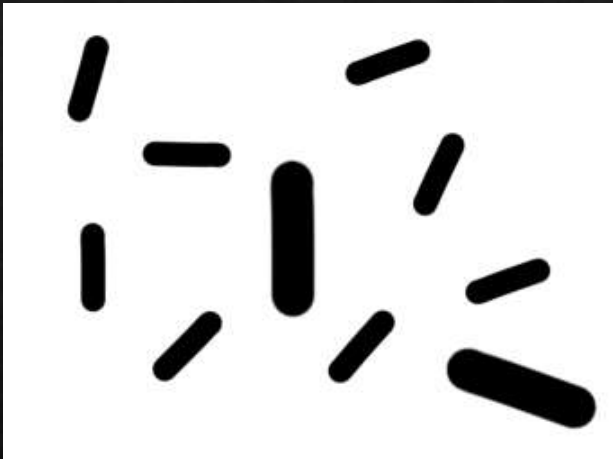
# OpenCV : Conclusion

0.8666666666666667	105426
0.8707070707070708	98734
0.8747474747474747	0
0.8787878787878788	0



# OpenCV : Conclusion

10 bâtons



10 bâtons

10 ronds

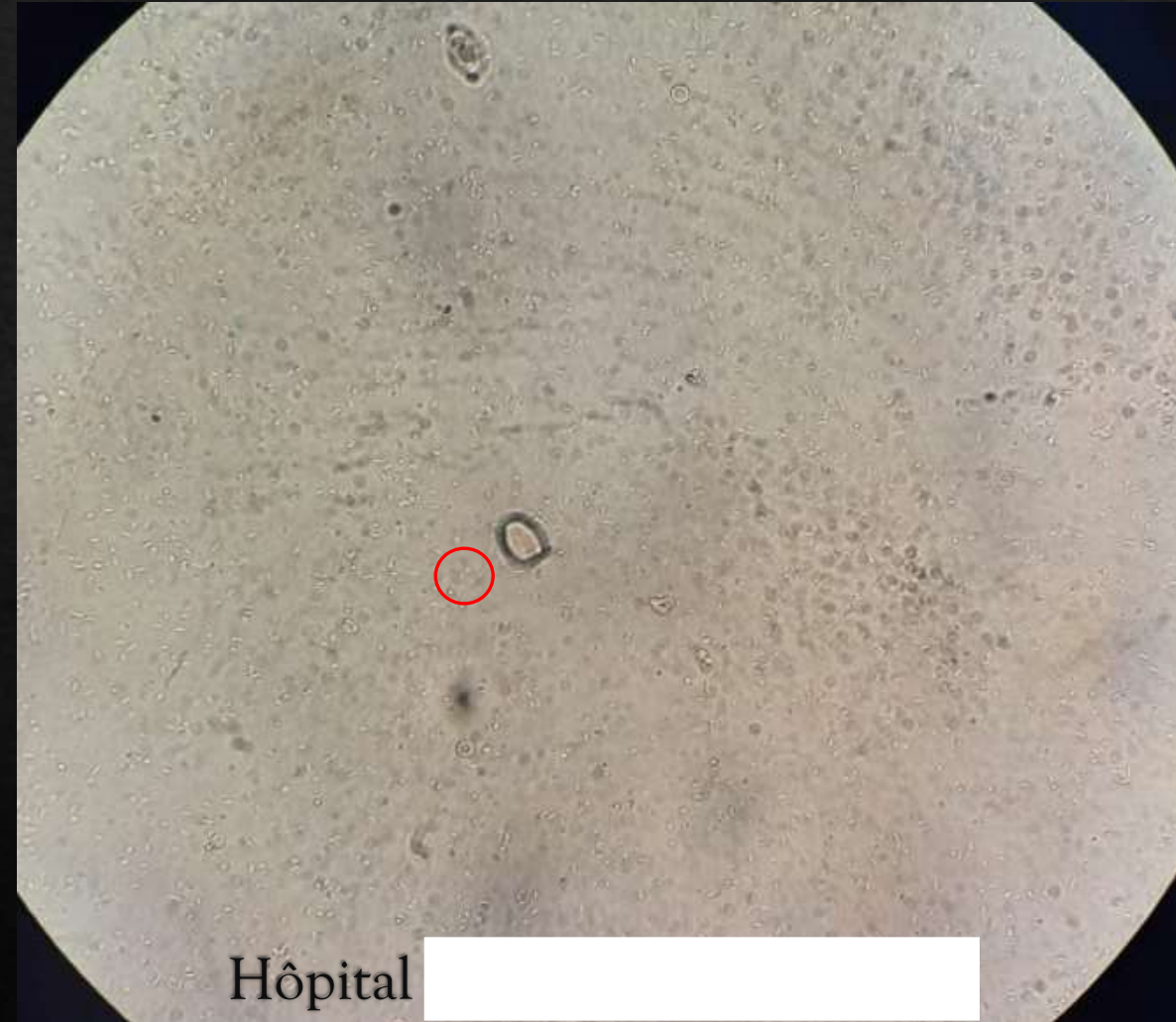
	A	B	C
1	seuillage	résultats1	résultats1
2	0.6	22	6
3	0.6222222222222222	22	4
4	0.6444444444444445	37	3
5	0.6666666666666666	37	2
6	0.6888888888888889	21	2
7	0.7111111111111111	21	2
8	0.7333333333333334	20	2
9	0.7555555555555555	20	2
10	0.7777777777777778	20	0
11	0.8	20	0
12		rond	baton

# Plan

- ◇ Mise en situation et description du problème
- ◇ Méthode mathématique
  - ◇ Introduction.
  - ◇ Etapes de la méthode.
  - ◇ Amélioration
  - ◇ Conclusion
- ◇ Utilisation de OpenCV
  - ◇ Introduction
  - ◇ Conclusion
- ◇ Comparaison des méthodes & conclusion




# Conclusion



Photos prise avec une camera posée  
sur l'oculaire d'un microscope .





Merci pour votre  
attention .

# Méthode mathématique : Partie code

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from scipy.interpolate import interp1d
5
6 def Zoom(image, facteur):
7     x = int(image.shape[0] * facteur)
8     y = int(image.shape[1] * facteur)
9     return cv2.resize(image, (x, y))
10
11 def Contours(img, valeur_seuillage=160):
12     image = cv2.imread(img, cv2.IMREAD_UNCHANGED)
13     image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
14     image_black = cv2.threshold(image_gray, valeur_seuillage, 255, cv2.THRESH_BINARY)[1]
15     contours = cv2.findContours(image_black, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)[0]
16     return contours
17
```

## Méthode mathématique : Partie code

```
18 def Pentes(contour, pas=1):
19     L = np.ones(0)
20     X=[]
21     Y=[]
22     for i in range(len(contour)):
23         X.append(contour[i][0][0])
24         Y.append(contour[i][0][1])
25     for i in range(0, len(contour), pas):
26         if contour[i][0][0] == contour[i - 1][0][0]:
27             if contour[i][0][1] - contour[i - 1][0][1] > 0:
28                 L = np.append(L, [500])
29             else:
30                 L = np.append(L, [-500])
31         else:
32             L = np.append(L, [(contour[i][0][1] - contour[i - 1][0][1]) / (contour[i][0][0] - contour[i - 1][0][0])])
33     return L
34
```

# Méthode mathématique : Partie code

```
35 def MoyGlissante(x, n):
36     if len(x) < n:
37         return str(n) + "est une valeur invalide"
38     L = np.ones(0)
39     s = 0
40     for i in range(len(x)):
41         if i < n - 1:
42             for j in range(i + 1):
43                 s += x[j]
44                 L = np.append(L, s / (i + 1))
45                 s = 0
46         else:
47             for j in range(i - n + 1, i + 1):
48                 s += x[j]
49                 L = np.append(L, s / n)
50                 s = 0
51     return L
52
```



# Méthode mathématique : Partie code

```
54 def diff(L, H):
55     S = 0
56     for i in range(len(L)):
57         S += abs(L[i] - H[i])
58     return S / len(L)
59
60
61 Y1=MoyGlissante(Pentes(Contours("rond.png",)[1]), 50)
62 Y2=MoyGlissante(Pentes(Contours("rond_1.png",)[1]), 50)
63 Y3=MoyGlissante(Pentes(Contours("Rond_1.png",)[1]), 50)
64 #Y1=Pentes(Contours("rond.png",)[1])
65 #Y2=Pentes(Contours("rond_defo.png",)[1])
66 #Y3=Pentes(Contours("rond_blur_light_texture.png",)[1])
67
68 X1=np.arange(len(Y1))
69 X2=np.arange(len(Y2))
70 X3=np.arange(len(Y3))
71
```

# Méthode mathématique : Partie code

```
72 y1=interp1d(X1,Y1,kind="linear")
73 y2=interp1d(X2,Y2,kind="linear")
74 y3=interp1d(X3,Y3,kind="linear")
75
76 l=max(len(Y1),len(Y2),len(Y3))
77
78 X1=np.linspace(0,len(Y1)-1,l)
79 X2=np.linspace(0,len(Y2)-1,l)
80 X3=np.linspace(0,len(Y3)-1,l)
81
82 new1=np.array([y1(x) for x in X1])
83 new2=np.array([y2(x) for x in X2])
84 new3=np.array([y3(x) for x in X3])
85
86 X1=np.arange(0,l,l/len(Y1))
87 X2=np.arange(0,l,l/len(Y2))
88 X3=np.arange(0,l,l/len(Y3))
89
90 plt.plot(X1,Y1,"r")
91 plt.plot(X2,Y2,'b')
92 plt.plot(X3,Y3,'y')
```

# Méthode mathématique : Partie code

```
57 def detection_rond(img, valeur_seuillage=160, Moy=50):
58     contours = Contours(img, valeur_seuillage)
59     rond=0
60     print("Le nombre de particules trouvé est : ", len(contours)-1)
61     Y1 = MoyGlissante(Pentes(Contours("rond_1.png", valeur_seuillage)[1]), Moy)
62     for contour in contours:
63         Y3= MoyGlissante(Pentes(contour), Moy)
64         X1 = np.arange(len(Y1))
65         X3 = np.arange(len(Y3))
66         y3 = interp1d(X3, Y3, kind="linear")
67         y1 = interp1d(X1, Y1, kind="linear")
68         l = max(len(Y1), len(Y3))
69         X1 = np.linspace(0, len(Y1) - 1, l)
70         X3 = np.linspace(0, len(Y3) - 1, l)
71         new1 = np.array([y1(x) for x in X1])
72         new3 = np.array([y3(x) for x in X3])
73         if diff(new3, new1)<125:
74             rond+=1
75     print("Le nombre de cocci dans l'image donnée est : ", rond)
76     return rond
```



## Méthode mathématique : Partie code

```
78 def detection_baton(img, valeur_seuillage=160, Moy=50):
79     contours = Contours(img, valeur_seuillage)
80     baton=0
81     Y1 = MoyGlissante(Pentes(Contours("baton_hauriz.png", valeur_seuillage)[1]), Moy)
82     for contour in contours:
83         Y3= MoyGlissante(Pentes(contour), Moy)
84         X1 = np.arange(len(Y1))
85         X3 = np.arange(len(Y3))
86         y3 = interp1d(X3, Y3, kind="linear")
87         y1 = interp1d(X1, Y1, kind="linear")
88         l = max(len(Y1), len(Y3))
89         X1 = np.linspace(0, len(Y1) - 1, l)
90         X3 = np.linspace(0, len(Y3) - 1, l)
91         new1 = np.array([y1(x) for x in X1])
92         new3 = np.array([y3(x) for x in X3])
93         if diff(new3, new1)<100:
94             baton+=1
95     print("Le nombre de bacilles dans l'image donnée est : ", baton)
96     return baton
```



## OpenCV : Partie code

```
5 def detection(X,Y,seuillage=0.6,weight=0.5):
6     image_globale = cv2.imread(Y, cv2.IMREAD_UNCHANGED)
7     image = cv2.imread(X, cv2.IMREAD_UNCHANGED)
8     result = cv2.matchTemplate(image_globale, image, cv2.TM_SQDIFF_NORMED)
9     positions = np.where(result >= seuillage)
10    positions1 = []
11    for i in range(len(positions[0])):
12        positions1.append((positions[1][i], positions[0][i]))
13    largeur_image = image.shape[1]
14    hauteur_image = image.shape[0]
15    rectangles = []
16    for pos in positions1:
17        rect = [int(pos[0]), int(pos[1]), largeur_image, hauteur_image]
18        rectangles.append(rect)
19        rectangles.append(rect)
20    rectangles, weights = cv2.groupRectangles(rectangles, 1, weight)
21    points = []
22    for (x, y, w, h) in rectangles:
23        points.append((x, y))
24        cv2.drawMarker(image_globale, (x, y), (255, 0, 255), 2, 40, 2)
25    cv2.waitKey()
26    return len(points)
```