



UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

COMPUTACION GRAFICA

Practica 12

Alumnos:

Chayña Batallanes Josnick
Perez Rodriguez Angelo Aldo
Pucho Zevallos Kelvin Paul
Vilcapaza Flores Luis Felipe
Sihuinta Perez Luis Armando

Julio 2021

Índice

	1
1. Algoritmo de computación gráfica Image Arithmetic (Adición y Sustracción de imágenes).	3
1.1. Implemente la adición de imágenes con los elementos de la Figura 1. Se recomiendo hacer un cast a la imagen antes del procesamiento para evitar el overflow en los píxeles.	3
1.2. Ahora implemente la adición con imágenes a colores (Figura 2).	5
1.3. Implemente la sustracción de imágenes para segmentar letras. Se le está brindando una foto del documento y otra de una hoja en blanco para eliminar el reflejo de la luz (Figura 3). Después de la sustracción aplique thresholding para obtener un resultado similar a la Figura 4. Tiene la libertad de aplicar métodos adicionales para mejorar los resultados, por ejemplo: contrast stretching, histogram equalization, etc.	7
1.4. Implemente la sustracción de imágenes para detectar el cambio o movimiento de objetos en fotogramas. En la Figura 5, se brindan dos fotogramas consecutivos, implemente la sustracción para obtener una imagen donde se visualiza qué objetos han cambiado de posición.	10
2. Algoritmo de computación gráfica Image Arithmetic (Multiplicación y División de imágenes)	12
2.1. Implemente la multiplicación de imagen por una constante con la Figura 1. Evalúe con $c = 2$, $c = 5$ y $c = 7$	12
2.2. Implemente la división de imágenes para segmentar letras. Se le está brindando una foto del documento y otra de una hoja en blanco para eliminar el reflejo de la luz (Figura 2). Después de la división debe normalizar la imagen a valores entre $[0 - 255]$ (Ecuación 1). Después aplique thresholding para obtener un resultado similar a la Figura 3. Tiene la libertad de aplicar métodos adicionales para mejorar los resultados, por ejemplo: contrast stretching, histogram equalization, etc.	14
2.3. Implemente la división de imágenes para detectar el cambio o movimiento de objetos en fotogramas. En la Figura 4, se brindan dos fotogramas consecutivos, implemente la división para obtener una imagen donde se visualiza que objetos se movieron, quizás sea necesario multiplicar el resultado por una constante para poder visualizar mejor los resultados. Después puede aplicar contrast stretching para mejorar aún más los resultados.	17
2.4. Implemente el operador Blending (ecuación 2) y evalúe sus resultados con imágenes de su preferencia, también pruebe diferentes valores de α	19
3. Algoritmo de computación gráfica Image Aritmética.(Review)	21
4. Algoritmo de computación gráfica Image Logical.	26
4.1. Implemente el operador AND con las imágenes de la Figura 1 para segmentar el objeto que aparece en ambas imágenes (intersección).	27

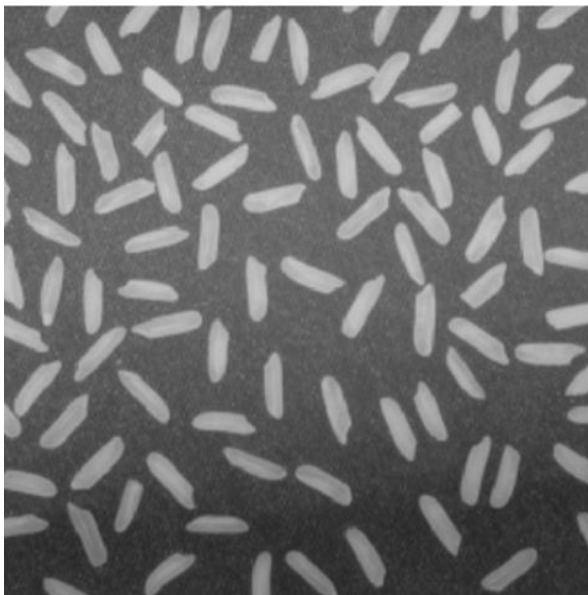
4.2. Implemente el operador OR con las imágenes de la Figura 1 para unir los objetos que aparecen en las imágenes (fusión).	30
4.3. Implemente el operador XOR con las imágenes de la Figura 1 para detectar los cambios.	34
5. Link de los códigos en github	38

1. Algoritmo de computación gráfica Image Arithmetic (Adición y Sustracción de imágenes).

1.1. Implemente la adición de imágenes con los elementos de la Figura 1. Se recomienda hacer un cast a la imagen antes del procesamiento para evitar el overflow en los píxeles.

■ Adiciones de Imágenes

• Imagen Original:



(a) Imagen 1



(b) Imagen 2

Figura 1: Adición de Imágenes

• Imagen Resultado :



■ Código:

```
1 import cv2
2 import numpy as np
3
4 imgReal1 = cv2.imread("image_1_1.jpg", cv2.IMREAD_GRAYSCALE) #8 bit
   por escala de grises
5 imgReal1 = cv2.resize(imgReal1, (400,400))
6 img1 = imgReal1
7
8 imgReal2 = cv2.imread("image_1_2.jpg", cv2.IMREAD_GRAYSCALE) #8 bit
   por escala de grises
9 imgReal2 = cv2.resize(imgReal2, (400,400))
10 img2 = imgReal2
11
12 cv2.imshow("Image Original 1", imgReal1)
13 cv2.imshow("Image Original 2", imgReal2)
14
15 img1 = img1.astype(int)
16 img2 = img2.astype(int)
17 imgResult = img1.copy()
18
19 for i in range(len(img1)):
20     for j in range(len(img1[0])):
21         sum_ = img1[i,j]//2 + img2[i,j]//2
22         if sum_ > 255 :
23             imgResult[i,j] = 255
24         else:
25             imgResult[i,j] = sum_
26 imgResult = imgResult.astype(np.uint8)
27 cv2.imshow("Resultado", imgResult)
28 filename = 'Resultado_Imagen1_2.jpg'
29 cv2.imwrite(filename, imgResult)
```

```
30  
31 cv2.waitKey(0)  
32 cv2.destroyAllWindows()
```

1.2. Ahora implemente la adición con imágenes a colores (Figura 2).

■ Adiciones de Imágenes

- Imagen Original:



(a) Imagen 1



(b) Imagen 2

Figura 2: Adición de Imágenes

- Imagen Resultado :



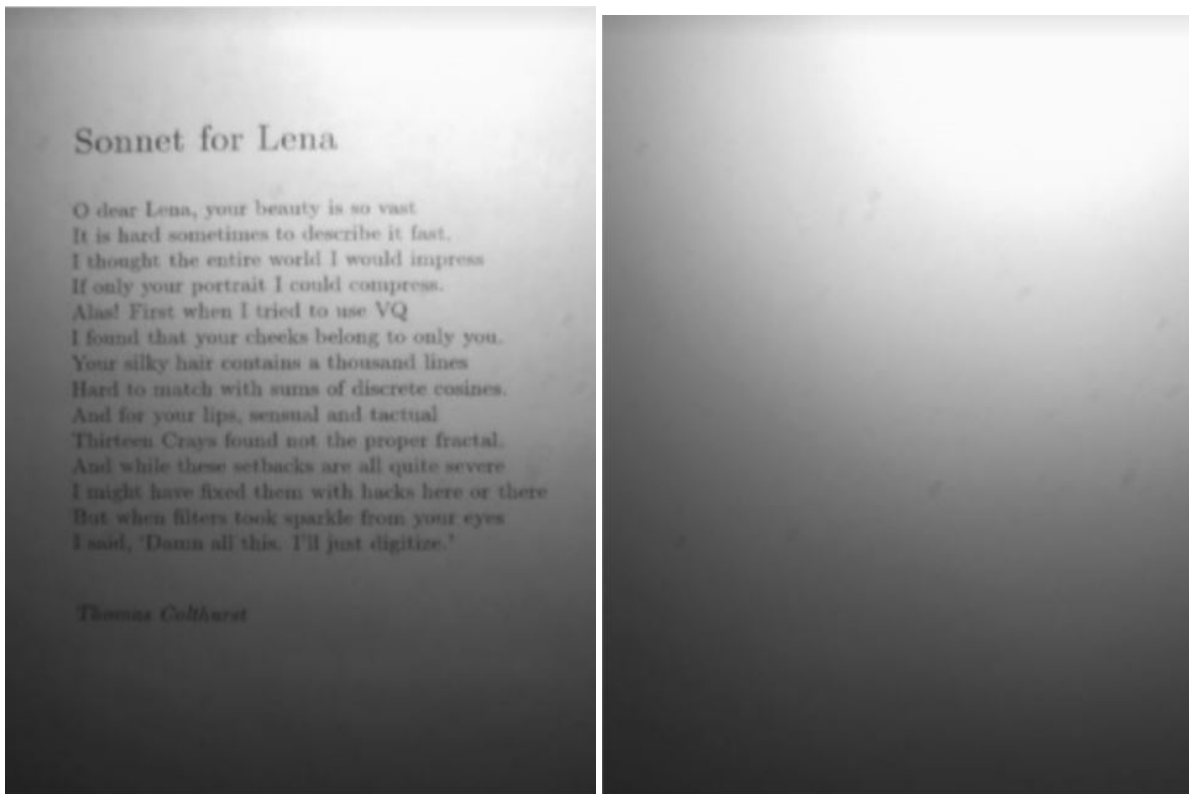
■ Código:

```
1 import cv2
2 import numpy as np
3
4 imgReal1 = cv2.imread("ciudad.jpg",1) #8 bit por escala de grises
5 imgReal1 = cv2.resize(imgReal1, (400,400))
6 img1 = imgReal1
7
8 imgReal2 = cv2.imread("leon.jpg",1) #8 bit por escala de grises
9 imgReal2 = cv2.resize(imgReal2, (400,400))
10 img2 = imgReal2
11
12
13 cv2.imshow("Image Original 1", imgReal1)
14 cv2.imshow("Image Original 2", imgReal2)
15
16 B1, G1, R1 = cv2.split(img1)
17 B2, G2, R2 = cv2.split(img2)
18 B1, G1, R1 = B1.astype(int), G1.astype(int), R1.astype(int)
19 B2, G2, R2 = B2.astype(int), G2.astype(int), R2.astype(int)
20 Bresult,Gresult,Rresult = B1.copy(), G1.copy(), R1.copy()
21
22 print(B1.dtype)
23 # print(img2)
24 for i in range(len(B1)):
25     for j in range(len(B1[0])):
26         sum_B = B1[i,j]//2 + B2[i,j]//2
27         if sum_B > 255 :
28             Bresult[i,j] = 255
29         else:
30             Bresult[i,j] = sum_B
31
32         sum_G = G1[i,j]//2 + G2[i,j]//2
33         if sum_G > 255 :
34             Gresult[i,j] = 255
35         else:
36             Gresult[i,j] = sum_G
37
38         sum_R = R1[i,j]//2 + R2[i,j]//2
39         if sum_R > 255 :
40             Rresult[i,j] = 255
41         else:
42             Rresult[i,j] = sum_R
43
44 img_result = cv2.merge((Bresult,Gresult,Rresult))
45 img_result = np.array(img_result, dtype=np.uint8)
46 cv2.imshow('ResultadoLeonCiudad', img_result)
47 filename = 'ResultadoLeonCiudad.jpg'
48 cv2.imwrite(filename, img_result)
49
50 cv2.waitKey(0)
51 cv2.destroyAllWindows()
```

- 1.3. Implemente la sustracción de imágenes para segmentar letras. Se le está brindando una foto del documento y otra de una hoja en blanco para eliminar el reflejo de la luz (Figura 3). Después de la sustracción aplique thresholding para obtener un resultado similar a la Figura 4. Tiene la libertad de aplicar métodos adicionales para mejorar los resultados, por ejemplo: contrast stretching, histogram equalization, etc.

■ Sustracción de Imágenes

- Imagen Original:



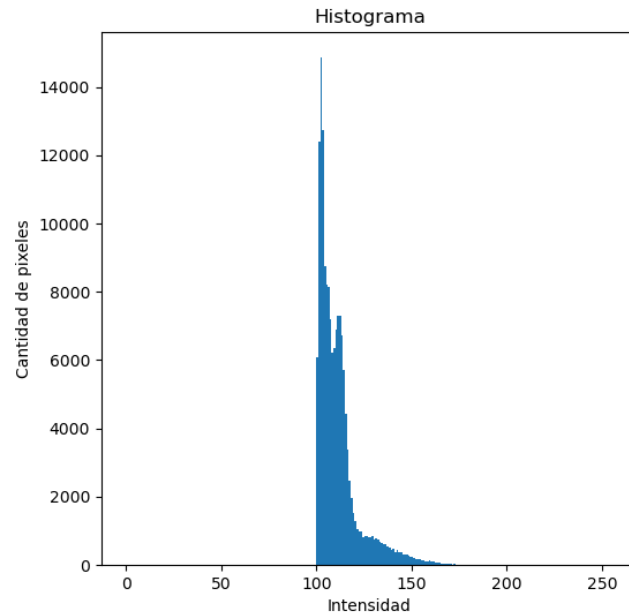
(a) Imagen 1

(b) Imagen 2

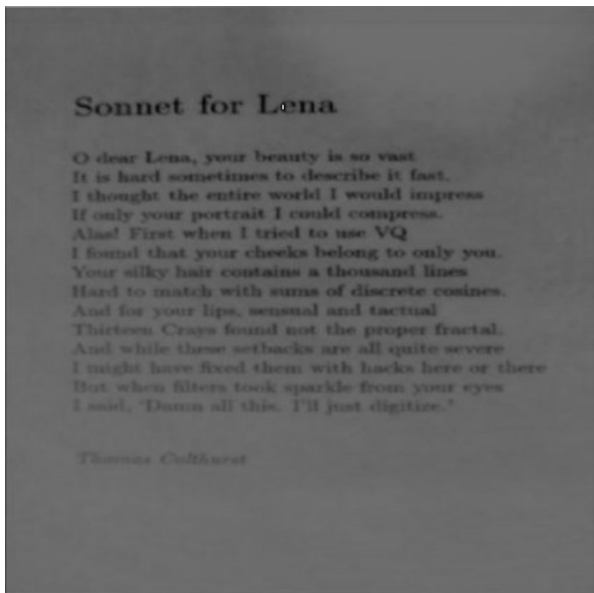
Figura 3: Sustracción de Imágenes

- Histograma :

Se uso un Thresold de 90



- Resultados :



(a) Imagen de la Sustracción de imágenes

Sonnet for Lena

O dear Lena, your beauty is so vast
 It is hard sometimes to describe it fast.
 I thought the entire world I would impress
 If only your portrait I could compress.
 Alas! First when I tried to use VQ
 I found that your cheeks belong to only you.
 Your silky hair contains a thousand lines
 Hard to match with sums of discrete cosines.
 And for your lips, sensual and tactual
 Thirteen Crays found not the proper fractal.
 And while these setbacks are all quite severe
 I might have fixed them with hacks here or there
 But when filters took sparkle from your eyes
 I said, "Damn all this. I'll just digitize."

Thomas Calhoun

(b) Imagen aplicando Thresolding

Figura 4: Sustracción de Imágenes

■ Código:

```
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
4
5
6 def getThresholding(img, threshold):
7     modifiedImg = img.copy()
8     print(img)
9     for i in range(len(img)):
10         for j in range(len(img[i])):
11             if(img[i][j] < threshold):
12                 modifiedImg[i][j] = np.uint8(0)
13             else:
14                 modifiedImg[i][j] = np.uint8(255)
15     return modifiedImg
16
17 imgReal1 = cv2.imread("texto1.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
18     escala de grises
19 imgReal1 = cv2.resize(imgReal1, (400,400))
20 img1 = imgReal1
21
22 imgReal2 = cv2.imread("textoblanco.jpg", cv2.IMREAD_GRAYSCALE) #8 bit
23     por escala de grises
24 imgReal2 = cv2.resize(imgReal2, (400,400))
25 img2 = imgReal2
26
27 cv2.imshow("Image Original 1", imgReal1)
28 cv2.imshow("Image Original 2", imgReal2)
29
30 img1 = img1.astype(int)
31 img2 = img2.astype(int)
32 imgResult = img1.copy()
33 # print(imgResult.dtype)
34 print(imgResult)
35 # imgResult = list(img1.flatten())
36 # imgResult = np.reshape(imgResult, (img1.shape[0], img1.shape[2]))
37
38 c = 110
39 for i in range(len(img1)):
40     for j in range(len(img1[0])):
41         rest = img1[i,j] - img2[i,j]
42         imgResult[i,j] = rest + c
43
44 imgResult = imgResult.astype(np.uint8)
45 cv2.imshow("Resultado Sustraccion", imgResult)
46
47 # filename = 'Resultado_Sustraccion.jpg'
48 # cv2.imwrite(filename, imgResult)
49
50 f, (ax1) = plt.subplots(1, 1, figsize=(6, 6))
51 ax1.hist(imgResult.ravel(), 256, [0,256])
52 ax1.set_title("Histograma")
53 ax1.set_xlabel('Intensidad')
54 ax1.set_ylabel('Cantidad de pixeles')
```

```

52
53 threshold = 90
54 imgThres = getThresholding(imgResult,threshold)
55 cv2.imshow("Resultado Thresolding", imgThres)
56
57 # filename = 'Resultado_Thresolding.jpg'
58 # cv2.imwrite(filename, imgThres)
59
60 # # Filename
61 # plt.savefig('HistogramaThresoldingSus.png')
62
63 plt.show()
64 cv2.waitKey(0)
65 cv2.destroyAllWindows()

```

1.4. Implemente la sustracción de imágenes para detectar el cambio o movimiento de objetos en fotogramas. En la Figura 5, se brindan dos fotogramas consecutivos, implemente la sustracción para obtener una imagen donde se visualiza qué objetos han cambiado de posición.

■ **Adiciones de Imágenes**

• **Imagen Original:**



(a) Imagen 1



(b) Imagen 2

Figura 5: Adición de Imágenes

• **Imagen Resultado :**



■ Código:

```
1  import cv2
2  import numpy as np
3  from matplotlib import pyplot as plt
4
5  imgReal1 = cv2.imread("tools_1.jpg", cv2.IMREAD_GRAYSCALE) #8 bit
   por escala de grises
6  imgReal1 = cv2.resize(imgReal1, (400,400))
7  img1 = imgReal1
8
9  imgReal2 = cv2.imread("tools_2.jpg", cv2.IMREAD_GRAYSCALE) #8 bit
   por escala de grises
10 imgReal2 = cv2.resize(imgReal2, (400,400))
11 img2 = imgReal2
12
13 cv2.imshow("Image Original 1", imgReal1)
14 cv2.imshow("Image Original 2", imgReal2)
15
16 img1 = img1.astype(int)
17 img2 = img2.astype(int)
18 imgResult = img1.copy()
19 # print(imgResult.dtype)
20 print(imgResult)
21
22 add = 100
23 for i in range(len(img1)):
24     for j in range(len(img1[0])):
25         rest = abs(img1[i,j] - img2[i,j])
26         imgResult[i,j] = rest
27 imgResult = imgResult.astype(np.uint8)
28 cv2.imshow("Resultado Sustraccion tools", imgResult)
29
```

```

30 filename = 'Resultado_Sustraccion_tools.jpg'
31 cv2.imwrite(filename, imgResult)
32
33 arrayImg = imgResult.ravel()
34 sortArray = np.sort(arrayImg)
35
36 # 0% y el 100 %
37 min_ = int(len(sortArray) * 0.0)
38 max_ = int(len(sortArray) * 1.0 - 1)
39 c = sortArray[min_]
40 d = sortArray[max_]
41
42 plt.show()
43 cv2.waitKey(0)
44 cv2.destroyAllWindows()

```

2. Algoritmo de computación gráfica Image Arithmetic (Multiplicación y División de imágenes)

2.1. Implemente la multiplicación de imagen por una constante con la Figura 1. Evalúe con $c = 2$, $c = 5$ y $c = 7$.

■ Multiplicación de Imágenes

• Imagen Original:



• Imagen Resultado :



(a) Imagen 1 con constante $c=2$ (b) Imagen 2 con constante $c=5$ (c) Imagen 3 con constante $c=7$

Figura 6: Multiplicación de Imágenes

■ Código:

```

1  import cv2
2  import numpy as np
3
4  imgReal1 = cv2.imread("tigre.jpg", 1) #8 bit por escala de grises
5  imgReal1 = cv2.resize(imgReal1, (400,400))
6  img1 = imgReal1
7
8  cv2.imshow("Image Original 1", imgReal1)
9
10 img1 = img1.astype(int)
11 constantes = [2,5,7]
12
13 B, G, R = cv2.split(img1)
14 Bresult,Gresult,Rresult = B.copy(), G.copy(), R.copy()
15 for c in constantes:
16     for i in range(len(img1)):
17         for j in range(len(img1[0])):
18
19             sum_B = B[i,j]*c
20             if sum_B > 255 :
21                 Bresult[i,j] = 255
22             else:
23                 Bresult[i,j] = sum_B
24
25             sum_G = G[i,j]*c
26             if sum_G > 255 :
27                 Gresult[i,j] = 255
28             else:
29                 Gresult[i,j] = sum_G
30
31             sum_R = R[i,j]*c
32             if sum_R > 255 :
33                 Rresult[i,j] = 255
34             else:
35                 Rresult[i,j] = sum_R
36

```

```

37     img_result = cv2.merge((Bresult,Gresult,Rresult))
38     img_result = np.array(img_result, dtype=np.uint8)
39     title_ = "Resultado con constante c = " + str(c)
40     cv2.imshow(title_, img_result)
41     # filename = "Resultado_C = "+str(c)+".jpg"
42     # cv2.imwrite(filename, img_result)
43
44     cv2.waitKey(0)
45     cv2.destroyAllWindows()

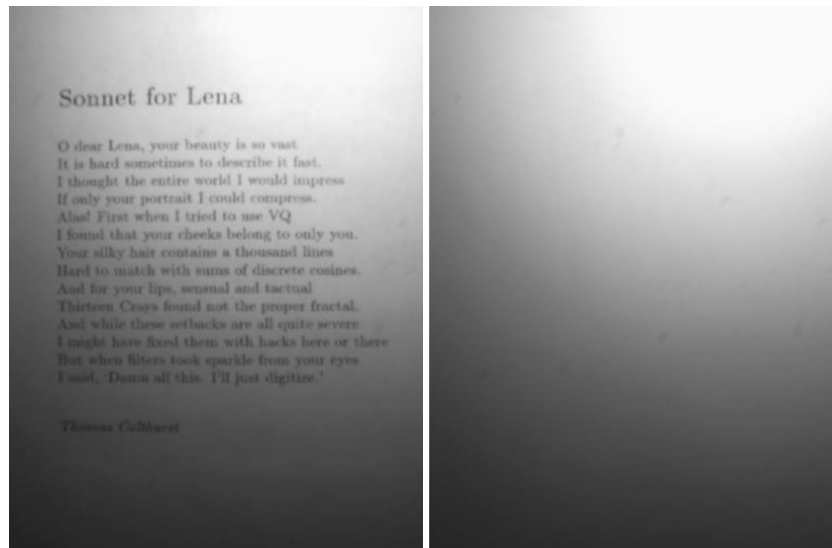
```

2.2. Implemente la división de imágenes para segmentar letras. Se le está brindando una foto del documento y otra de una hoja en blanco para eliminar el reflejo de la luz (Figura 2). Después de la división deber a normalizar la imagen a valores entre [0 – 255] (Ecuación 1). Después aplique thresholding para obtener un resultado similar a la Figura 3. Tiene la libertad de aplicar métodos adicionales para mejorar los resultados, por ejemplo: contrast stretching, histogram equalization, etc.

Después de la división se procedió a normalizar la imagen a valores entre [0 – 255] con la siguiente ecuación

■ División de Imágenes

• Imagen Original:

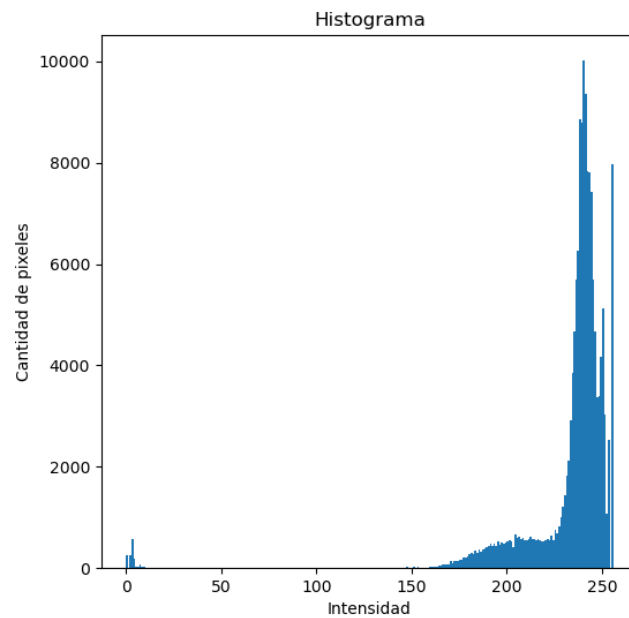


(a) Imagen 1

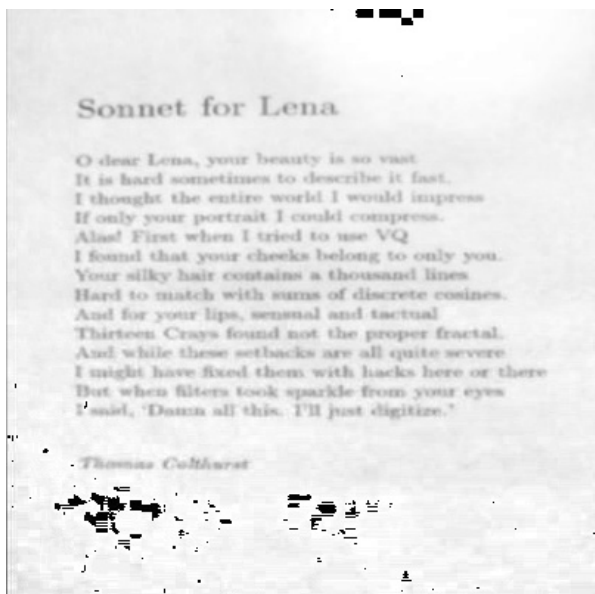
(b) Imagen 2

Figura 7: División de Imágenes

- Histograma para aplicar un Thresold:



- Imagen Resultado :



Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

Thomas Colthurst

(a) Resultado de la división de las imágenes normalizado

(b) Resultado final después de aplicar Thresolding

Figura 8: Resultados Finales

- Código:


```

1  import cv2
2  import numpy as np
3  from matplotlib import pyplot as plt
4  import math
5
6  def getThresholding(img,threshold):
7      modifiedImg = img.copy()
8      # print(img)
9      for i in range(len(img)):
10         for j in range(len(img[i])):
11             if(img[i][j] > 135 and img[i][j] < 222):
12                 modifiedImg[i][j] = np.uint8(0)
13             else:
14                 modifiedImg[i][j] = np.uint8(255)
15         return modifiedImg
16
17  imgReal1 = cv2.imread("text1.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
18     escala de grises
19  imgReal1 = cv2.resize(imgReal1, (400,400))
20  img1 = imgReal1
21
22  imgReal2 = cv2.imread("text2.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
23     escala de grises
24  imgReal2 = cv2.resize(imgReal2, (400,400))
25  img2 = imgReal2
26
27  cv2.imshow("Image Original 1", imgReal1)
28  cv2.imshow("Image Original 2", imgReal2)
29
30  img1 = img1.astype(int)
31  img2 = img2.astype(int)
32  imgResult = img1.copy()
33
34  for i in range(len(img1)):
35      for j in range(len(img1[0])):
36          result = img1[i,j]/img2[i,j]
37          scala = (result-0)*(255-0)/(1-0) + 0
38          imgResult[i,j] = scala
39  imgResult = imgResult.astype(np.uint8)
40  cv2.imshow("Resultado Division text", imgResult)
41
42  # filename = "Resultado_Division_text.jpg"
43  # cv2.imwrite(filename, imgResult)
44
45  f, (ax1) = plt.subplots(1, 1,figsize=(6, 6))
46  ax1.hist(imgResult.ravel(),256,[0,256])
47  ax1.set_title("Histograma")
48  ax1.set_xlabel('Intensidad')
49  ax1.set_ylabel('Cantidad de pixeles')
50
51  threshold = 120
52  imgThres = getThresholding(imgResult,threshold)
53  cv2.imshow("Resultado Thresolding", imgThres)

```

```

54 # filename = "Resultado_Thresolding_text.jpg"
55 # cv2.imwrite(filename, imgThres)
56
57
58 # # Filename
59 # plt.savefig('HistogramaThresoldingtext.png')
60
61 plt.show()
62 cv2.waitKey(0)
63 cv2.destroyAllWindows()

```

2.3. Implemente la división de imágenes para detectar el cambio o movimiento de objetos en fotogramas. En la Figura 4, se brindan dos fotogramas consecutivos, implemente la división para obtener una imagen donde se visualiza que objetos se movieron, quizás sea necesario multiplicar el resultado por una constante para poder visualizar mejor los resultados. Después puede aplicar contrast stretching para mejorar a ´un mas los resultados .

■ **División de Imágenes**

• **Imagen Original:**

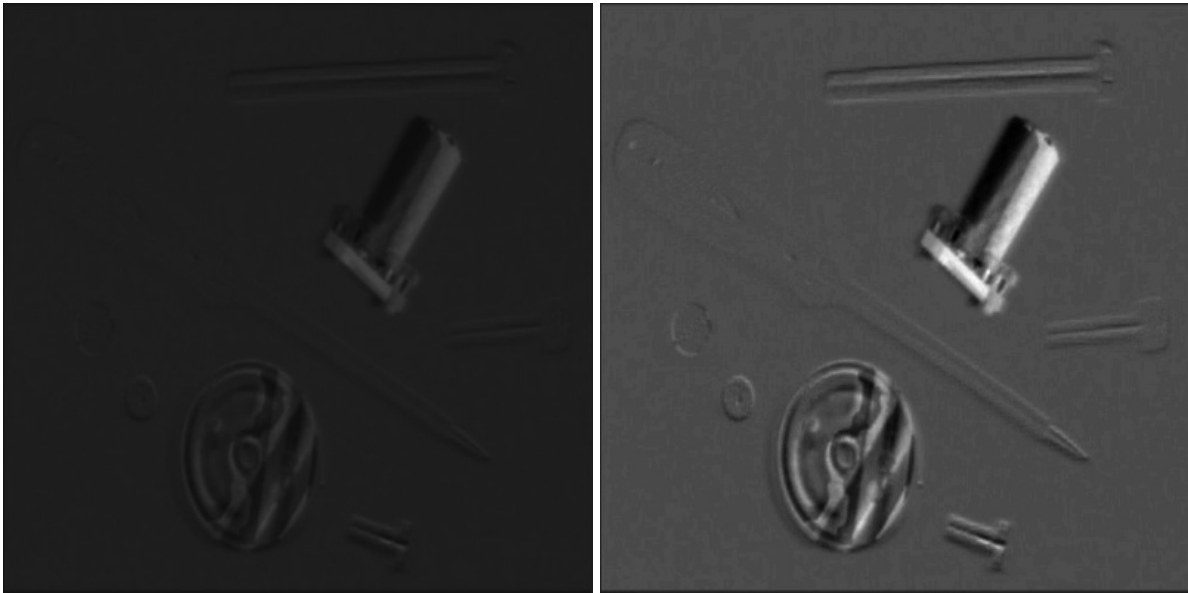


(a) Imagen 1

(b) Imagen 2

Figura 9: División de Imágenes

• **Imagen Resultado :**



(a) Resultado de la división de las imágenes multiplicado por 30
(b) Resultado final después de aplicar Contrast Stretching

Figura 10: Resultados Finales

■ Código:

```

1  import cv2
2  import numpy as np
3  import math
4
5  def PixelOperations(img,c,d):
6      a = 0
7      b = 255
8      for row in range(len(img)):
9          for pixel in range(len(img[row])):
10             if(0<=img[row][pixel] and img[row][pixel] <= c):
11                 img[row][pixel] = (a/c)*img[row][pixel]
12             elif(c<img[row][pixel] and img[row][pixel] <= d):
13                 img[row][pixel] = ((img[row][pixel] - c) * ((b-a)/(d-
14                     c)))+a
15             else:
16                 img[row][pixel] = ((img[row][pixel] - d) * ((255-b)
17                     / (255-d)))+b
18
19         img = np.array(img, dtype=np.uint8)
20         return img
21
22     imgReal1 = cv2.imread("tools1.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
23         escala de grises
24     imgReal1 = cv2.resize(imgReal1, (400,400))
25     img1 = imgReal1
26
27     imgReal2 = cv2.imread("tools2.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
28         escala de grises

```

```

24 imgReal2 = cv2.resize(imgReal2, (400,400))
25 img2 = imgReal2
26
27 cv2.imshow("Image Original 1", imgReal1)
28 cv2.imshow("Image Original 2", imgReal2)
29
30 img1 = img1.astype(int)
31 img2 = img2.astype(int)
32 imgResult = img1.copy()
33
34 for i in range(len(img1)):
35     for j in range(len(img1[0])):
36         result = (img2[i,j]/img1[i,j])*30
37         # scala = (result-0)*(255-0)/(1-0) + 0
38         imgResult[i,j] = result
39 imgResult = imgResult.astype(np.uint8)
40 cv2.imshow("Resultado", imgResult)
41
42 filename = "Resultado_Division_Tools.jpg"
43 cv2.imwrite(filename, imgResult)
44
45 arrayImg = imgResult.ravel()
46 sortArray = np.sort(arrayImg)
47 # print(sortArray)
48 # 0% y el 100 %
49 min_ = int(len(sortArray) * 0.0)
50 max_ = int(len(sortArray) * 1.0 -1)
51 c = sortArray[min_]
52 d = sortArray[max_]
53 img_Constrast = PixelOperations(imgResult.astype(int),int(c),int(d))
54 cv2.imshow("Resultado contrast", img_Constrast)
55
56 filename = "Resultado_contrast_Tools.jpg"
57 cv2.imwrite(filename, img_Constrast)
58
59 cv2.waitKey(0)
60 cv2.destroyAllWindows()

```

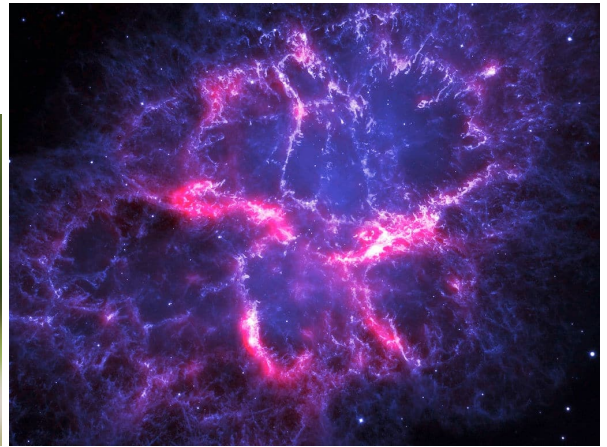
2.4. Implemente el operador Blending (ecuación 2) y evalúe sus resultados con imágenes de su preferencia, también pruebe diferentes valores de X.

■ Blending de Imágenes

- Imagen Original:



(a) Imagen 1



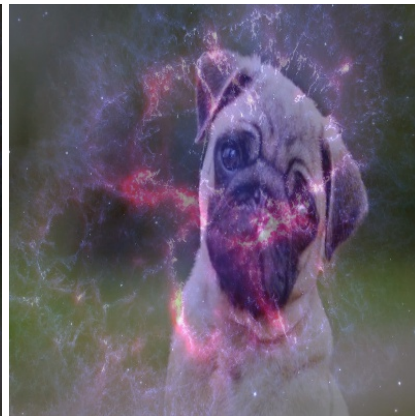
(b) Imagen 2

Figura 11: Blending

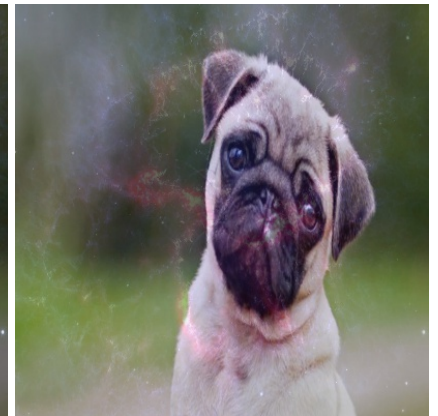
• **Imagen Resultado :**



(a) Imagen 1 con constante $X=0.25$



(b) Imagen 2 con constante $X=0.5$



(c) Imagen 3 con constante $X=0.75$

Figura 12: Resultado despues de aplicar Blending con diferentes valores X

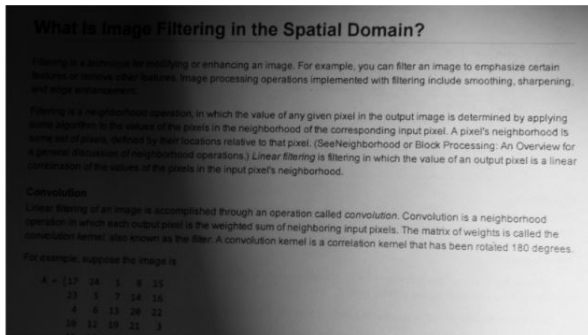
■ Código:

```
1 import cv2
2 import numpy as np
3
4 imgReal1 = cv2.imread("pug.jpg",1) #8 bit por escala de grises
5 imgReal1 = cv2.resize(imgReal1,(400,400))
6 img1 = imgReal1
7
8 imgReal2 = cv2.imread("magico.jpg",1) #8 bit por escala de grises
9 imgReal2 = cv2.resize(imgReal2,(400,400))
10 img2 = imgReal2
11
12 cv2.imshow("Image Original 1", imgReal1)
13 cv2.imshow("Image Original 2", imgReal2)
14
15 B1, G1, R1 = cv2.split(img1)
16 B2, G2, R2 = cv2.split(img2)
17 B1, G1, R1 = B1.astype(int), G1.astype(int), R1.astype(int)
18 B2, G2, R2 = B2.astype(int), G2.astype(int), R2.astype(int)
19 Bresult,Gresult,Rresult = B1.copy(), G1.copy(), R1.copy()
20
21 X = [0.25,0.5,0.75]
22 for x in X:
23     for i in range(len(B1)):
24         for j in range(len(B1[0])):
25
26             Bresult[i,j] = x*B1[i, j] + (1-x)*B2[i, j]
27             Gresult[i,j] = x*G1[i, j] + (1-x)*G2[i, j]
28             Rresult[i,j] = x*R1[i, j] + (1-x)*R2[i, j]
29
30     img_result = cv2.merge((Bresult,Gresult,Rresult))
31     img_result = np.array(img_result, dtype=np.uint8)
32     title_ = "Resultado_con_constante_C_"+str(x)
33     cv2.imshow(title_, img_result)
34
35     # filename = "Resultado_con_constante_C_"+str(x)+".jpg"
36     # cv2.imwrite(filename, img_result)
37
38 cv2.waitKey(0)
39 cv2.destroyAllWindows()
```

3. Algoritmo de computación gráfica Image Aritmética.(Review)

■ Imágenes Originales

- Imagen Original:



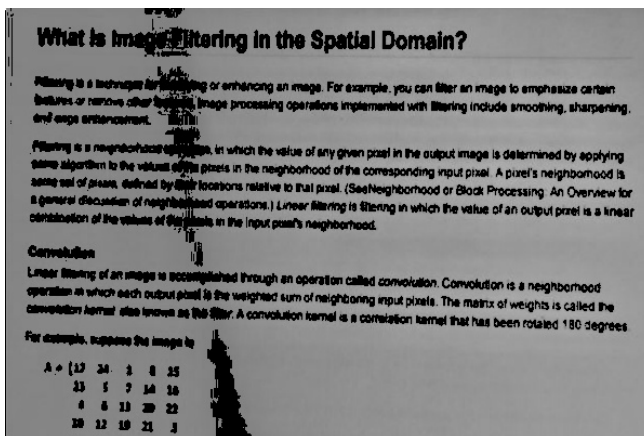
(a) Imagen 1



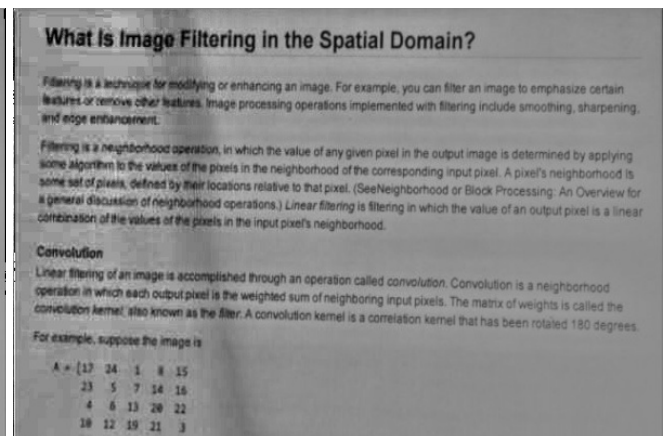
(b) Imagen 2

Figura 13: Blending

- **Resultado después de aplicar Sustracción y División de imágenes :**



(a) Resultado de la sustracción

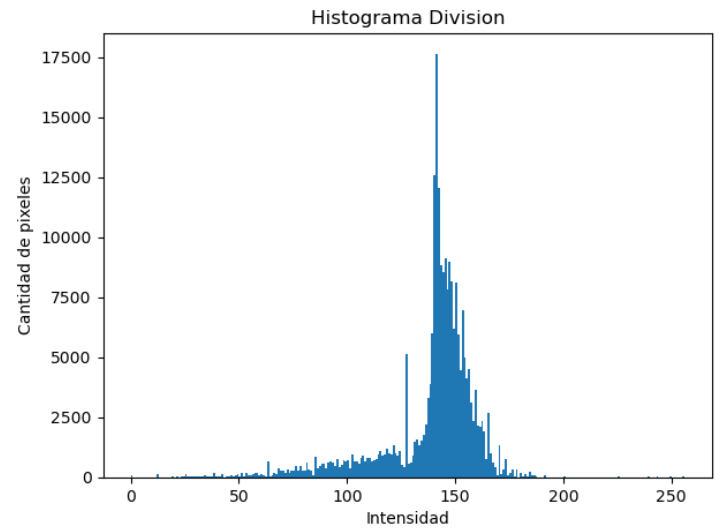
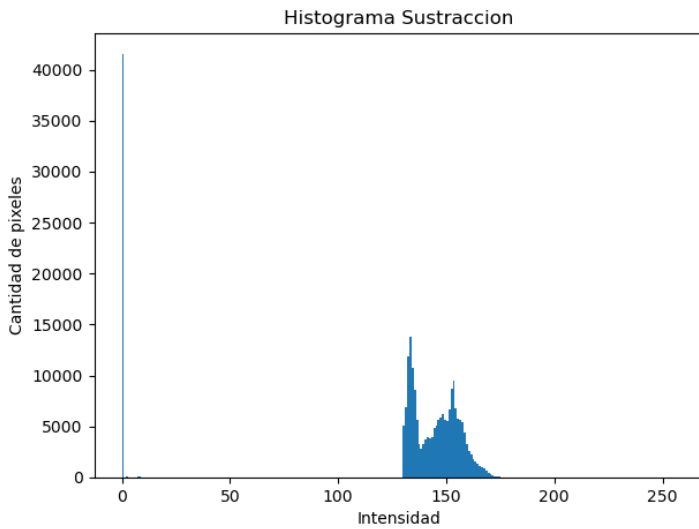


(b) Resultado de la división

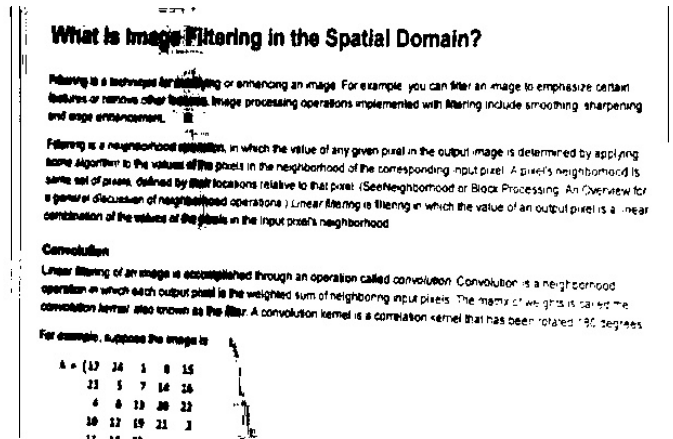
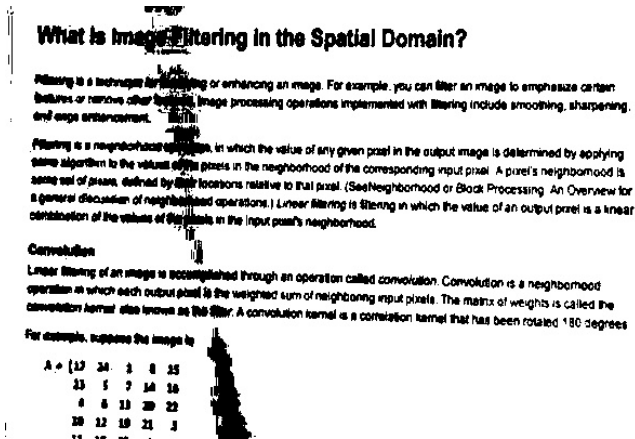
Figura 14: Resultado después de aplicar Sustracción y División

- **Histogramas :**

Para la sustracción de un Thresold de 1 y para la división se uso un Thresold de 130 pero hubo una condición más para tratar con el problema de la mancha que se ven en ambas imagenes. La condicion tenia un rango de 116 a 129 donde se trato de desaparecer dicha mancha pero tambien se borro algunas palabras por lo que no se pudo borrar del todo. Se presentó una opción donde se trata recuperar las palabras y que la mancha quede un poco gris para no perjudicar palabras.



• Resultados :



(a) Resultado de la sustracción después de aplicar Thresolding (b) Resultado de la división después de aplicar Thresolding

Figura 15: Resultado después de aplicar Sustracción y División con un Thresold

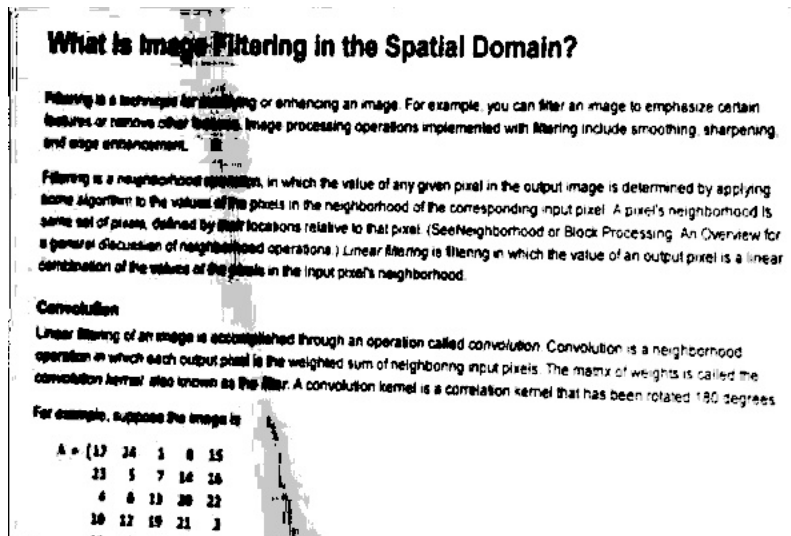


Figura 16: Resultado después de aplicar Sustracción y División con un Threshold

■ Código:

```

1  import cv2
2  import numpy as np
3  from matplotlib import pyplot as plt
4
5
6  def getThresholding(img, threshold):
7      modifiedImg = img.copy()
8      # print(img)
9      for i in range(len(img)):
10         for j in range(len(img[i])):
11             if(img[i][j] < threshold):
12                 modifiedImg[i][j] = np.uint8(0)
13             else:
14                 modifiedImg[i][j] = np.uint8(255)
15         return modifiedImg
16
17  def getThresholding2(img):
18      modifiedImg = img.copy()
19      # print(img)
20      for i in range(len(img)):
21         for j in range(len(img[i])):
22             if(img[i][j] >= 116 and img[i][j] <= 129):
23                 modifiedImg[i][j] = np.uint8(255) #200
24             elif(img[i][j] < 130):
25                 modifiedImg[i][j] = np.uint8(0)
26             else:
27                 modifiedImg[i][j] = np.uint8(255)
28         return modifiedImg
29
30  imgReal1 = cv2.imread("textReview1.jpg", cv2.IMREAD_GRAYSCALE) #8 bit
31  por escala de grises
32  imgReal1 = cv2.resize(imgReal1, (600,400))

```

```

32 img1 = imgReal1.copy()
33
34 imgReal2 = cv2.imread("textReview2.jpg", cv2.IMREAD_GRAYSCALE) #8 bit
    por escala de grises
35 imgReal2 = cv2.resize(imgReal2, (600,400))
36 img2 = imgReal2.copy()
37
38 cv2.imshow("Image Original 1", imgReal1)
39 cv2.imshow("Image Original 2", imgReal2)
40
41 img1 = img1.astype(int)
42 img2 = img2.astype(int)
43 imgResultSust = img1.copy()
44 imgResultDiv = img1.copy()
45
46 # =====
47 # SUSTRACION
48 # =====
49 c = 130
50 for i in range(len(img1)):
51     for j in range(len(img1[0])):
52         rest = img1[i,j] - img2[i,j]
53         if(rest <0):
54             imgResultSust[i,j] = 0
55         else:
56             scala = rest + c
57             if(scala >255):
58                 imgResultSust[i,j] = 0
59             else:
60                 imgResultSust[i,j] = rest + c
61
62 imgResultSust = imgResultSust.astype(np.uint8)
63 cv2.imshow("Resultado Sustraccion", imgResultSust)
64 # filename = 'ReviewSustraccion.jpg'
65 # cv2.imwrite(filename, imgResultSust)
66
67 f, (ax1,ax2) = plt.subplots(1, 2,figsize=(15, 5))
68 ax1.hist(imgResultSust.ravel(),256,[0,256])
69 ax1.set_title("Histograma Sustraccion")
70 ax1.set_xlabel('Intensidad')
71 ax1.set_ylabel('Cantidad de pixeles')
72
73 threshold = 1
74 imgSustThres = getThresholding(imgResultSust,threshold)
75 cv2.imshow("Thresolding Sustraccion", imgSustThres)
76 # filename = 'ReviewThresoldingSust.jpg'
77 # cv2.imwrite(filename, imgSustThres)
78
79 # =====
80 # DIVISION
81 # =====
82
83 for i in range(len(img1)):
84     for j in range(len(img1[0])):
85         result = (img1[i,j]/img2[i,j])
86         # print(result)

```

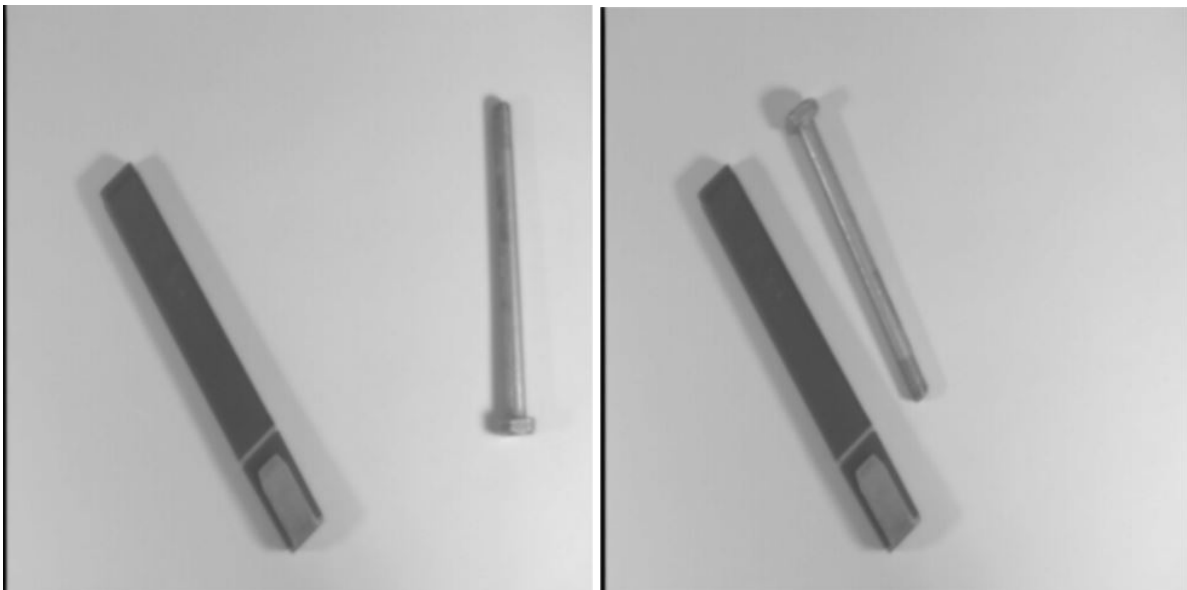
```

87         scala = (result-0)*(255-0)/(2-0) + 0
88         imgResultDiv[i,j] = int(scala)
89     imgResultDiv = imgResultDiv.astype(np.uint8)
90     cv2.imshow("Resultado Division", imgResultDiv)
91     # filename = 'ReviewDivision.jpg'
92     # cv2.imwrite(filename, imgResultDiv)
93
94     ax2.hist(imgResultDiv.ravel(),256,[0,256])
95     ax2.set_title("Histograma Division")
96     ax2.set_xlabel('Intensidad')
97     ax2.set_ylabel('Cantidad de pixeles')
98
99     imgDivThres = getThresholding2(imgResultDiv)
100    cv2.imshow("Thresolding Division", imgDivThres)
101    # filename = 'ReviewThresoldingDiv.jpg'
102    # cv2.imwrite(filename, imgDivThres)
103
104    # Filename
105    plt.savefig('HistogramaRview.png')
106    plt.show()
107    cv2.waitKey(0)
108    cv2.destroyAllWindows()

```

4. Algoritmo de computación gráfica Image Logical.

■ Imágenes de Pruebas



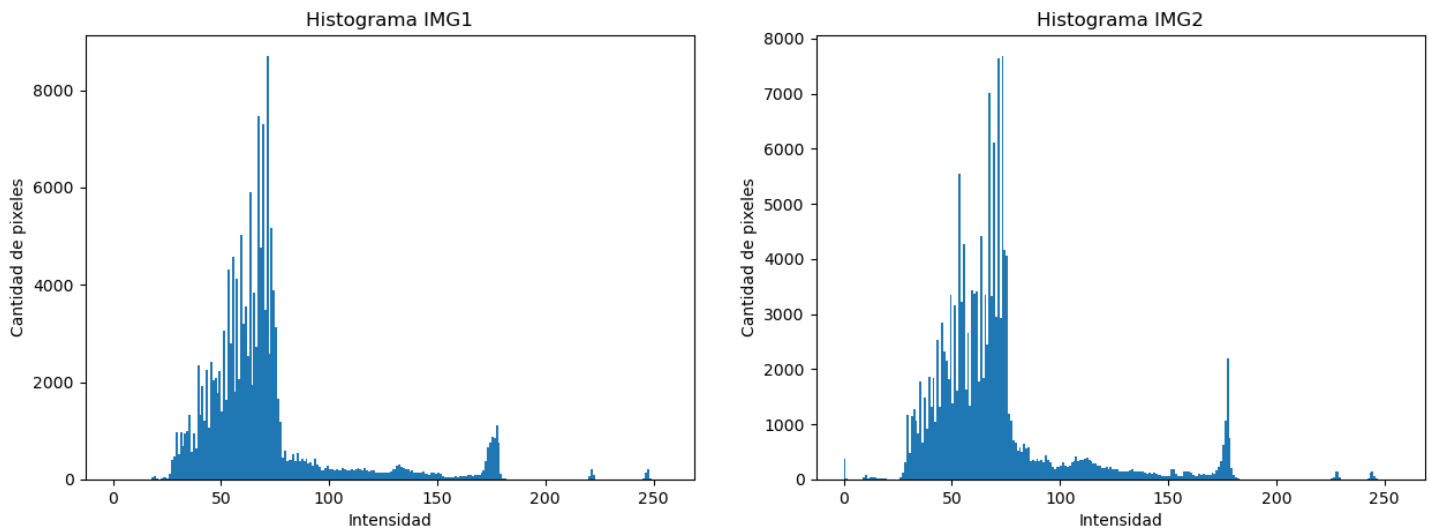
(a) Imagen 1

(b) Imagen 2

Figura 17: Blending

4.1. Implemente el operador AND con las imágenes de la Figura 1 para segmentar el objeto que aparece en ambas imágenes (intersección).

- Histograma para analizar y definir un Threshold
Se usó un threshold de 98



- Imágenes después de invertir los colores

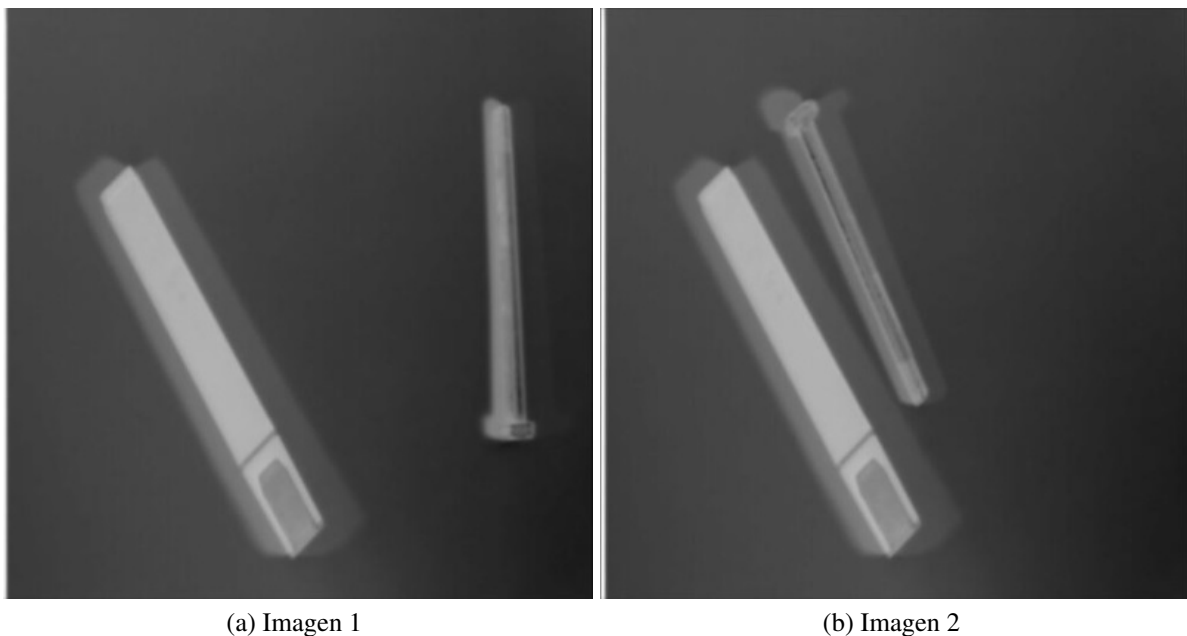


Figura 18: Inversión de colores en las imágenes

- Imágenes después de aplicar Thresholding
- Resultados después de aplicar el operador AND

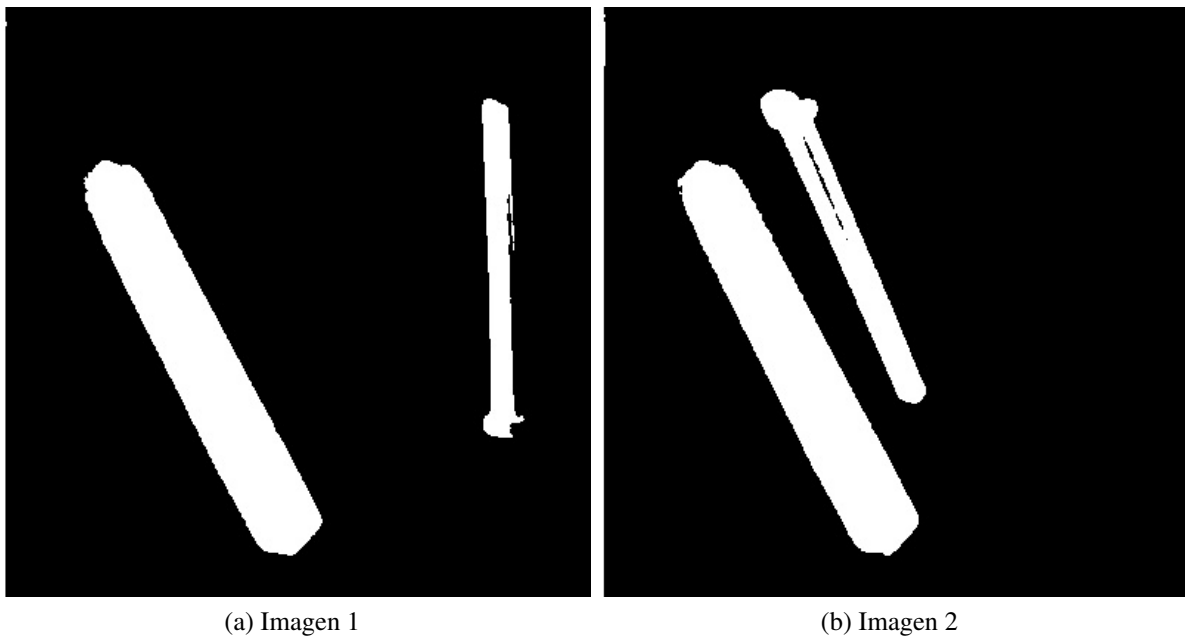
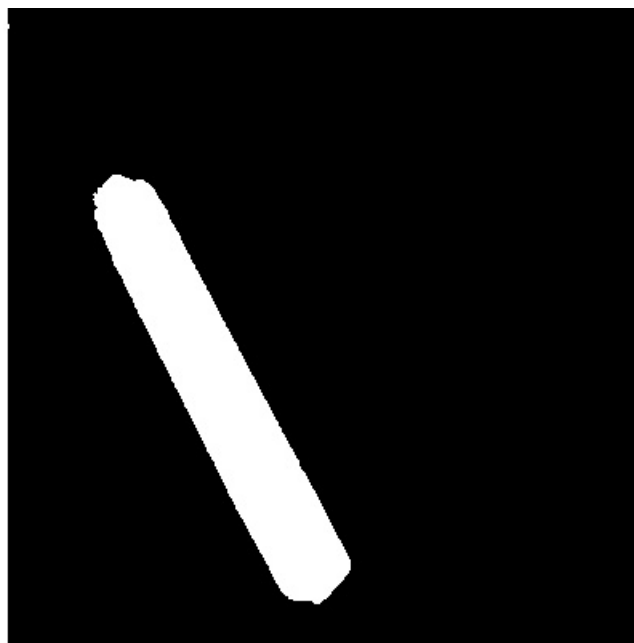


Figura 19: Thresholding

Al aplicar el operador AND a cada pixel de las imágenes en su forma binaria se puede llegar a un resultado en la cual se muestra en la figura. La figura resultante muestra que las dos imágenes tienen un objeto en común.



■ Código:

```
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
4
5
6 def getThresholding(img,threshold):
7     modifiedImg = img.copy()
8     # print(img)
9     for i in range(len(img)):
10         for j in range(len(img[i])):
11             if(img[i][j] < threshold):
12                 modifiedImg[i][j] = np.uint8(0)
13             else:
14                 modifiedImg[i][j] = np.uint8(255)
15     return modifiedImg
16
17 imgReal1 = cv2.imread("pernos1.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
18     escala de grises
19 imgReal1 = cv2.resize(imgReal1, (400,400))
20 img1 = imgReal1.copy()
21
22 imgReal2 = cv2.imread("pernos2.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
23     escala de grises
24 imgReal2 = cv2.resize(imgReal2, (400,400))
25 img2 = imgReal2.copy()
26
27
28 img1 = 255-img1
29 img2 = 255-img2
30
31 cv2.imshow("Image Original 1", img1)
32 cv2.imshow("Image Original 2", img2)
33
34
35 f, (ax1,ax2) = plt.subplots(1, 2,figsize=(15, 5))
36 ax1.hist(img1.ravel(),256,[0,256])
37 ax1.set_title("Histograma IMG1")
38 ax1.set_xlabel('Intensidad')
39 ax1.set_ylabel('Cantidad de pixeles')
40
41 ax2.hist(img2.ravel(),256,[0,256])
42 ax2.set_title("Histograma IMG2")
43 ax2.set_xlabel('Intensidad')
44 ax2.set_ylabel('Cantidad de pixeles')
45
46
47
48 threshold = 98
49 imgThres1 = getThresholding(img1,threshold)
50 cv2.imshow("Resultado Thresolding de la IMG1", imgThres1)
51 # filename = 'ResultadoThresoldingPernos1.jpg'
52 # cv2.imwrite(filename, imgThres1)
53
54
55 imgThres2 = getThresholding(img2,threshold)
56 cv2.imshow("Resultado Thresolding de la IMG2", imgThres2)
57 # filename = 'ResultadoThresoldingPernos2.jpg'
58 # cv2.imwrite(filename, imgThres2)
```

```

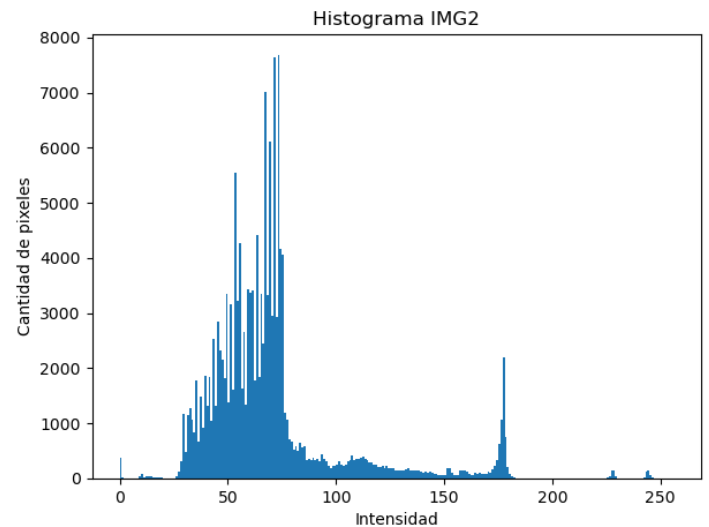
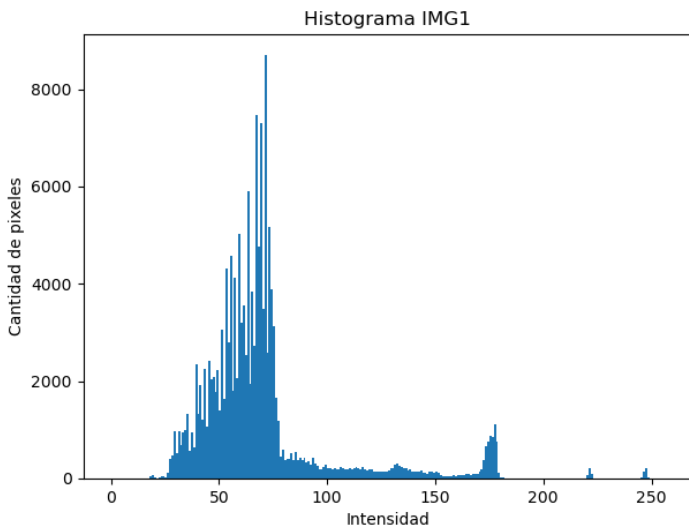
52
53 # =====
54 # OPERADOR AND
55 # =====
56 imgResult = imgThres1.copy()
57
58 for i in range(len(imgThres1)):
59     for j in range(len(imgThres1[0])):
60         result = imgThres1[i,j] & imgThres2[i,j]
61         imgResult[i,j] = result
62 cv2.imshow("Resultado Operador AND", imgResult)
63
64 # filename = 'ResultadoOperadorAND.jpg'
65 # cv2.imwrite(filename, imgResult)
66
67 # Filename
68 # plt.savefig('HistogramaThresoldPernos.png')
69
70 plt.show()
71 cv2.waitKey(0)
72 cv2.destroyAllWindows()

```

4.2. Implemente el operador OR con las imágenes de la Figura 1 para unir los objetos que aparecen en las imágenes (fusión).

■ Histograma para analizar y definir un Thresold

Se uso un thresold de 98



■ **Imagenes despues despues de invertir los colores**

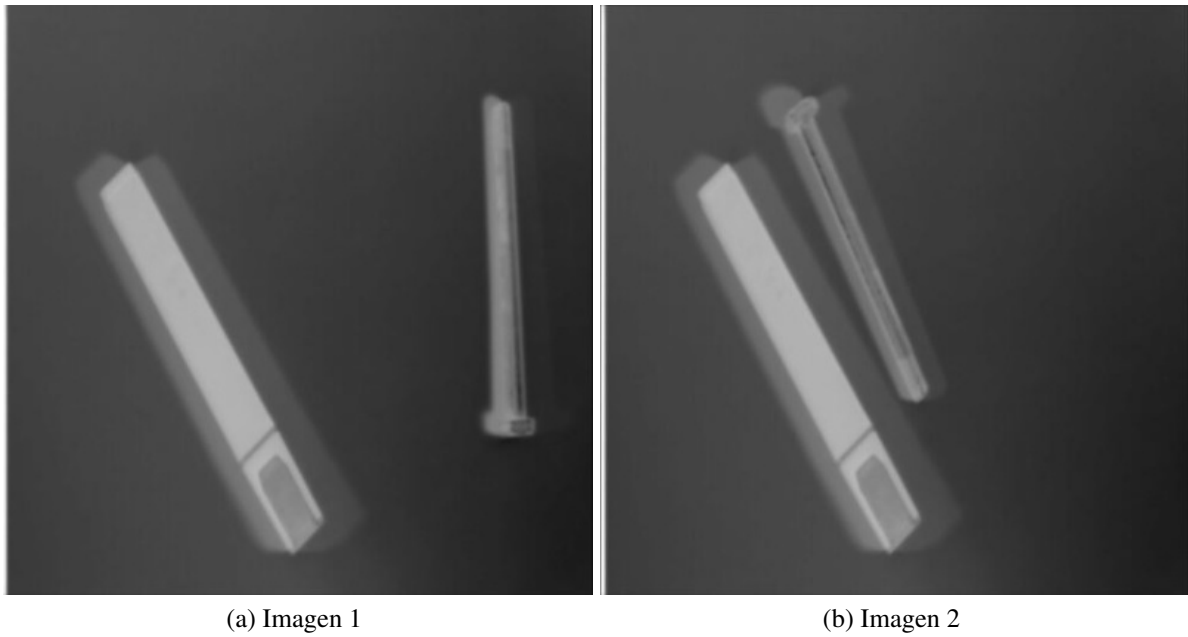


Figura 20: Invercion de colores en las imagenes

■ **Imagenes despues de aplicar Thresolding**

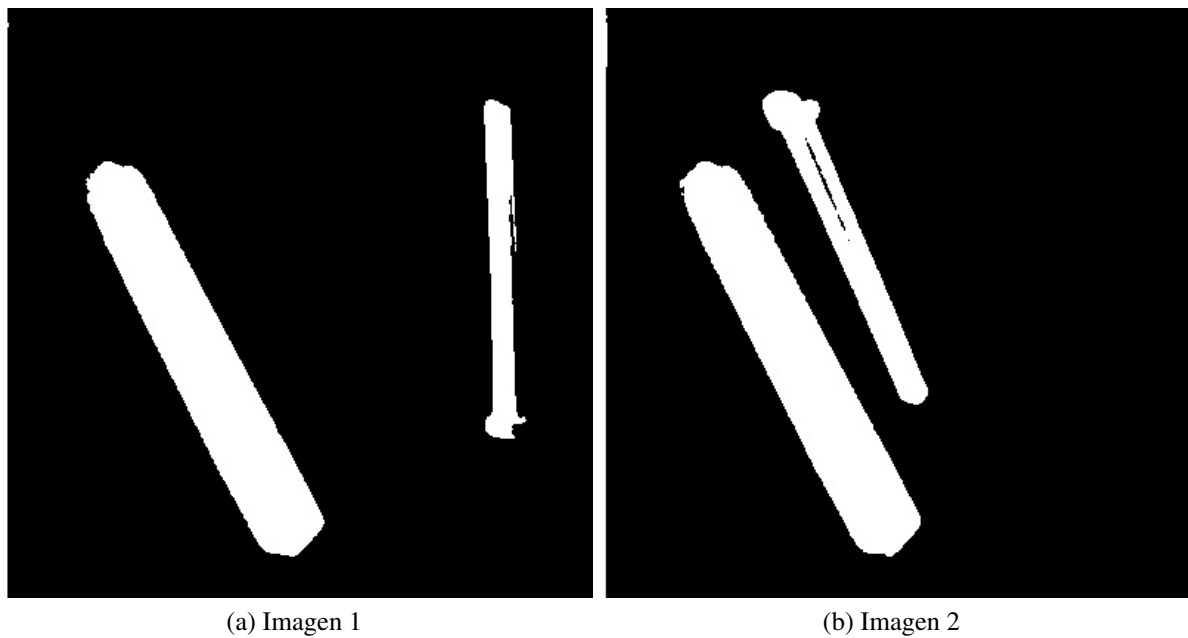
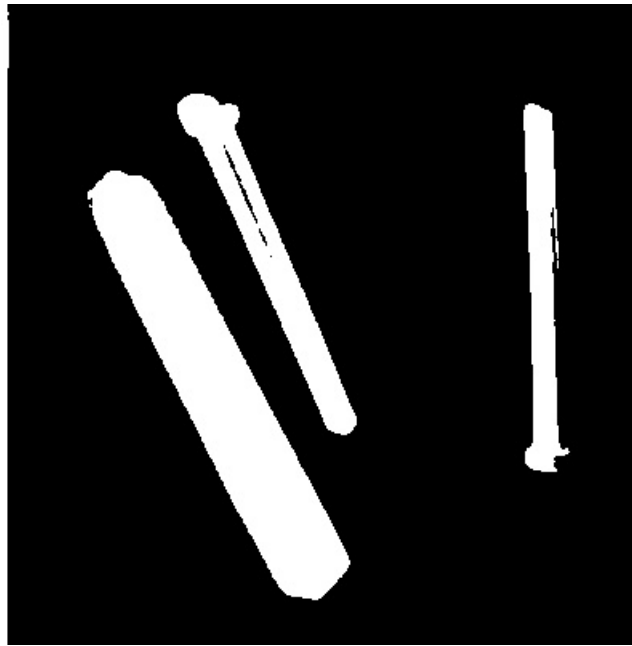


Figura 21: Thresolding

- Resultados despues de aplicar el operador OR



Al aplicar el operador OR a cada pixel de las imagenes en su forma binaria se puede llegar a un resultado en la cual se muestra en la figura. La figura resultante muestra que las dos imagene pueden llegar a unirse.

- **Codigo:**

```
1  import cv2
2  import numpy as np
3  from matplotlib import pyplot as plt
4
5
6  def getThresholding(img,threshold):
7      modifiedImg = img.copy()
8      # print(img)
9      for i in range(len(img)):
10         for j in range(len(img[i])):
11             if(img[i][j] < threshold):
12                 modifiedImg[i][j] = np.uint8(0)
13             else:
14                 modifiedImg[i][j] = np.uint8(255)
15         return modifiedImg
16
17  imgReal1 = cv2.imread("pernos1.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
18     escala de grises
19  imgReal1 = cv2.resize(imgReal1, (400,400))
20  img1 = imgReal1.copy()
21
22  imgReal2 = cv2.imread("pernos2.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
23     escala de grises
24  imgReal2 = cv2.resize(imgReal2, (400,400))
```

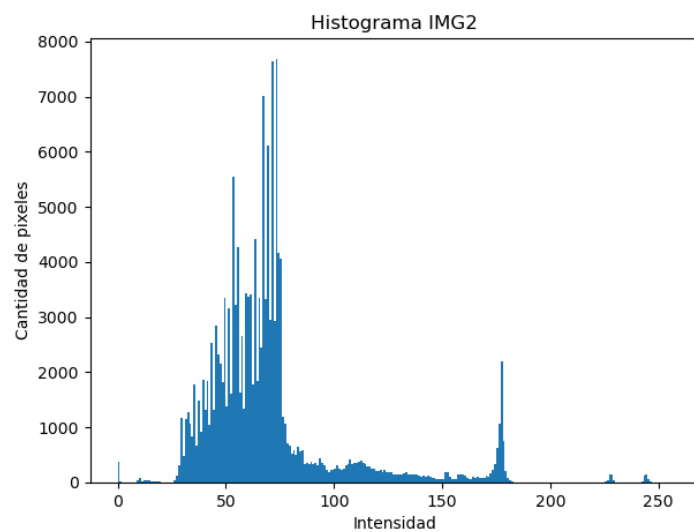
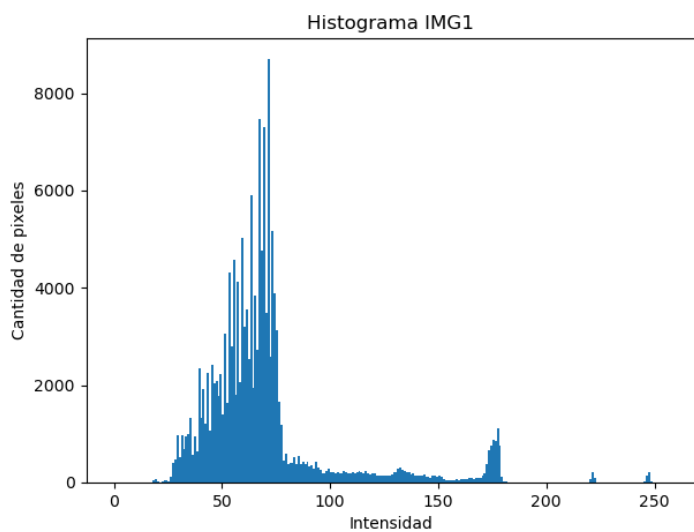
```

23  img2 = imgReal2.copy()
24
25  img1 = 255-img1
26  img2 = 255-img2
27
28
29  cv2.imshow("Image Original 1", img1)
30  cv2.imshow("Image Original 2", img2)
31
32  f, (ax1,ax2) = plt.subplots(1, 2,figsize=(15, 5))
33  ax1.hist(img1.ravel(),256,[0,256])
34  ax1.set_title("Histograma IMG1")
35  ax1.set_xlabel('Intensidad')
36  ax1.set_ylabel('Cantidad de pixeles')
37  ax2.hist(img2.ravel(),256,[0,256])
38  ax2.set_title("Histograma IMG2")
39  ax2.set_xlabel('Intensidad')
40  ax2.set_ylabel('Cantidad de pixeles')
41
42
43  threshold = 98
44  imgThres1 = getThresholding(img1,threshold)
45  cv2.imshow("Resultado Thresolding de la IMG1", imgThres1)
46
47  imgThres2 = getThresholding(img2,threshold)
48  cv2.imshow("Resultado Thresolding de la IMG2", imgThres2)
49
50
51  # =====
52  # OPERADOR OR
53  # =====
54
55  imgResult = imgThres1.copy()
56
57  for i in range(len(imgThres1)):
58      for j in range(len(imgThres1[0])):
59          result = imgThres1[i,j] | imgThres2[i,j]
60          imgResult[i,j] = result
61  cv2.imshow("Resultado Operador OR", imgResult)
62  filename = 'ResultadoOperadorOR.jpg'
63  cv2.imwrite(filename, imgResult)
64
65  plt.show()
66  cv2.waitKey(0)
67  cv2.destroyAllWindows()

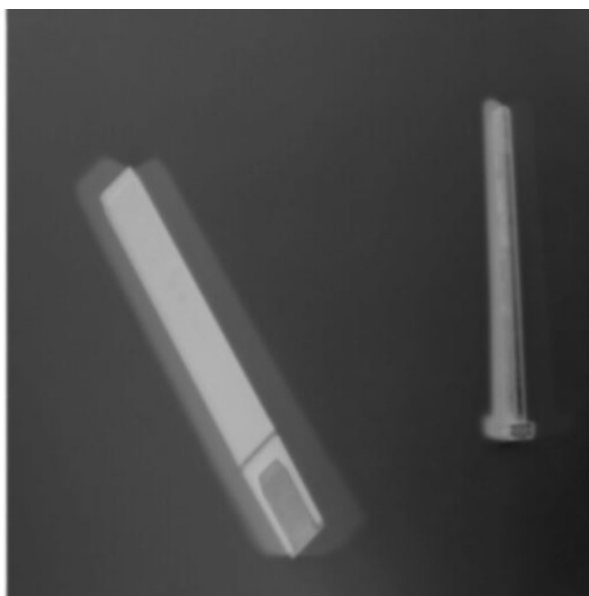
```

4.3. Implemente el operador XOR con las imágenes de la Figura 1 para detectar los cambios.

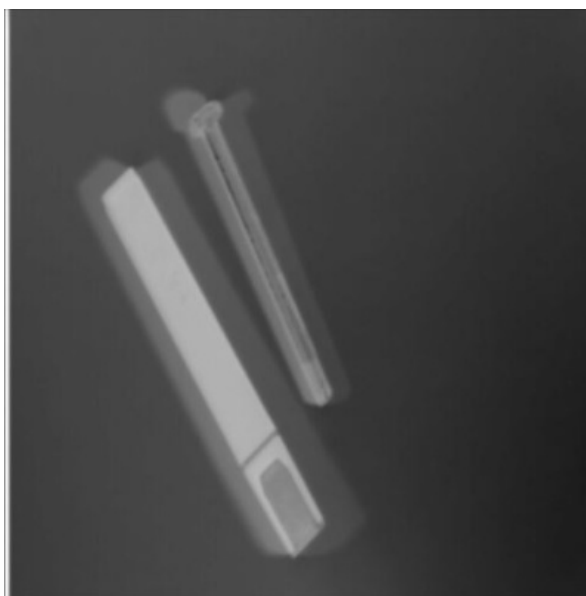
- Histograma para analizar y definir un Thresold
Se uso un thresold de 98



- Imagenes despues despues de invertir los colores



(a) Imagen 1



(b) Imagen 2

Figura 22: Invercion de colores en las imagenes

- Imagenes despues de aplicar Thresolding

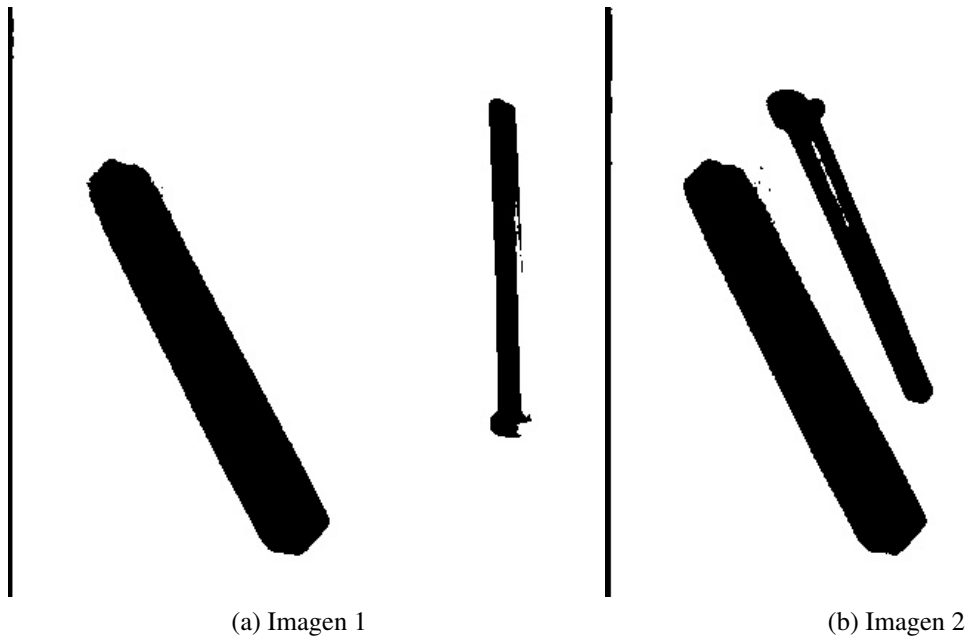
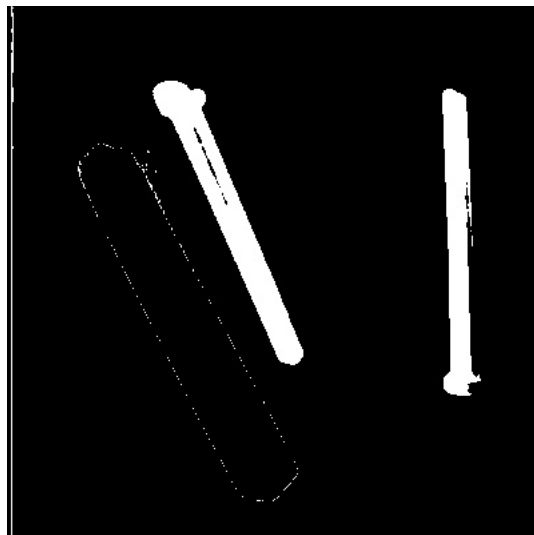


Figura 23: Thresholding

- Resultados despues de aplicar el operador XOR

Al aplicar el operador OR a cada pixel de las imagenes en su forma binaria se puede llegar a un resultado en la cual se muestra en la figura. La figura resultante muestra al objeto que tuvo ciertos cambios dando como resultado el grafico de los cambios del dicho objeto



■ Código:

```
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
4
5
6 def getThresholding(img,threshold):
7     modifiedImg = img.copy()
8     # print(img)
9     for i in range(len(img)):
10         for j in range(len(img[i])):
11             if(img[i][j] < threshold):
12                 modifiedImg[i][j] = np.uint8(255)
13             else:
14                 modifiedImg[i][j] = np.uint8(0)
15     return modifiedImg
16
17 imgReal1 = cv2.imread("pernos1.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
18     escala de grises
19 imgReal1 = cv2.resize(imgReal1, (400,400))
20 img1 = imgReal1.copy()
21
22 imgReal2 = cv2.imread("pernos2.jpg", cv2.IMREAD_GRAYSCALE) #8 bit por
23     escala de grises
24 imgReal2 = cv2.resize(imgReal2, (400,400))
25 img2 = imgReal2.copy()
26
27
28 img1 = 255-img1
29 img2 = 255-img2
30
31
32 cv2.imshow("Image Original 1", img1)
33 cv2.imshow("Image Original 2", img2)
34
35
36
37 f, (ax1,ax2) = plt.subplots(1, 2,figsize=(15, 5))
38 ax1.hist(img1.ravel(),256,[0,256])
39 ax1.set_title("Histograma IMG1")
40 ax1.set_xlabel('Intensidad')
41 ax1.set_ylabel('Cantidad de pixeles')
42 ax2.hist(img2.ravel(),256,[0,256])
43 ax2.set_title("Histograma IMG2")
44 ax2.set_xlabel('Intensidad')
45 ax2.set_ylabel('Cantidad de pixeles')
46
47
48
49 threshold = 96
50 imgThres1 = getThresholding(img1,threshold)
51 cv2.imshow("Resultado Thresolding de la IMG1", imgThres1)
52 # filename = 'ResultadoThresoldingPernos1_2.jpg'
53 # cv2.imwrite(filename, imgThres1)
54
55
56 imgThres2 = getThresholding(img2,threshold)
57 cv2.imshow("Resultado Thresolding de la IMG2", imgThres2)
58 # filename = 'ResultadoThresoldingPernos2_2.jpg'
```

```

52 # cv2.imwrite(filename, imgThres2)
53
54
55 # =====
56 # OPERADOR XOR
57 # =====
58
59 imgResult = imgThres1.copy()
60
61 for i in range(len(imgThres1)):
62     for j in range(len(imgThres1[0])):
63         result = imgThres1[i,j] ^ imgThres2[i,j]
64         imgResult[i,j] = result
65 cv2.imshow("Resultado Operador XOR", imgResult)
66 # filename = 'ResultadoOperadorXOR.jpg'
67 # cv2.imwrite(filename, imgResult)
68
69 plt.show()
70 cv2.waitKey(0)
71 cv2.destroyAllWindows()

```

5. Link de los códigos en github

`https://github.com/kpzaolod6000/Graphics-Computing/tree/main/
parcial_3/lab_10`