TRATAMIENTO DE SOMBRAS EN FOTOGRAFIAS DE DOCUMENTOS

```
# importamos librerias utilizadas
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage, misc
import matplotlib.pyplot as plt
# definimos unafuncion para graficar las imagenes
def imprimir pic(pic,etiqueta):
    plt.figure()
    plt.imshow(pic, cmap = 'gray')
    plt.title(etiqueta)
    plt.axis('off')
Seleccionamos la imagen con la cual se va a trabajar
imagen entrada = "input8"
input pic = cv2.imread(imagen entrada+".jpeg")
METODO BASE
Generamos la mascara o "perfil de sombra"
# Paquetes necesarios para la morfología matemática
from skimage.morphology import erosion, dilation, opening, closing
# Elementos estructurales
from skimage.morphology import disk, diamond, ball, rectangle, star
from scipy import ndimage as ndi
#reducimos la pic a grises
input pic = cv2.cvtColor(input pic, cv2.COLOR BGR2GRAY)
#realizamos una copia de la pic recibida para poder modificarla en un
espacio de memoria diferente
mask = np.copy(input pic)
#las sisguientes 3 lineas se encargar de normalizar la pic de 0 a 255
mask = mask - mask.min()
mask = mask/mask.max()
mask = mask*255
#Creamos la mascara con un filtro de mediana, consideramos las letras
como ruido
```

for i in range(3):

```
mask = ndimage.median_filter(mask, size=18,mode='reflect')
mask_gris = np.copy(mask)

imprimir_pic(mask,'Máscara en grises')

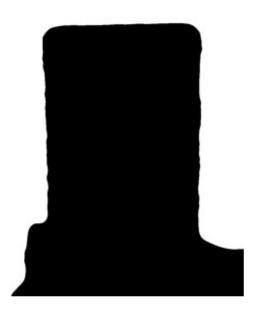
#la maskara debe estar en blanco y negro y no en escala de grises, asi
que se realiza un if:
promedio = np.mean(mask)
for (cor_y,cor_x), value in np.ndenumerate(mask):
    if mask[cor_y,cor_x]promedio: #Good Pixel
        mask[cor_y,cor_x] = 0.1
    elif mask[cor_y,cor_x]>promedio: #Bad Pixel
        mask[cor_y,cor_x] = 254.9

imprimir_pic(mask,'Máscara')
imprimir_pic(input_pic,'original')
```

Máscara en grises



Máscara



original

y = 0.0 X = 3, Y = 0.0 X = 4, Y = 0.0 X = 5, Y = 0. Por lo tanto P(U = 0) = 0,28. Y = 0.0 X = 3, Y = 0.0 X = 5, Y = 0. Por lo tanto P(U = 0) = 0,28. Be caso de las probabilidades associadas con U se pueden obtener de una mamera integiate. Por tanto, la distribución de probabilidades de U se puede resumir caso ingo: u: 0,1,2,3; P(U = 0): 0,23; 0,00; 0,25; 0,17. La distribución de probabilidades de las variables aleatorias V y W como se definió anteriormente se tabilidades de las variables bidimensional continua y si Z = H₁(X, Y) es una insocio continua de (X, Y), cutonices Z será una variable aleatoria continua (unidenzazional) y el problema de encontrar un (de se algo más complicado. Para resobre este problema acessissames un tecumis que vainos a indicasa; discutir a continuación. Antés de baccio, hosquejemos beresenante la idea básica.

Para encontrar la fig de Z = H₁(X, Y) es a menulu mas implie introducir una seganda variable aleatoria, digamos W = H₂(X, Y), y obtenes primiero la fide conjunta de Z y W, digiamas glir), al integrar implemente k(z, w) con respectó a w. Eno es.

g(z) = [** léz, w) dw.

Los problemas que quedas son (1) cônte encostrar la fide conjunta de Z y W, di dilino problema, simplemente, digamos que generalmente lacrono is elección intermediario, y realmante no son intercas, el viciliano problema. (2 y W necutiamos el sucressa. 4).

Teoremes 4.3. Sepocagosas que (1, Y) en una variable abaciera la life encontrar and continua con fife consumo. (3 secretas (3 secretas de 1) mana. Con el fin de encontrar na continua con fife consumo. (3 secretas (4 secretas de 1) mana. Con el fin de encontrar na continua con fife consumo. (4 secretas proposale abaccion de 1) de contento continua con fife consumo.

```
Definimos la funcion de contraste que apartir de un percentil permite
def funcion de contraste(pic,percentil):
    limite = np.percentile(pic,percentil)
    blancos = pic >= limite
    pic = pic + 255*blancos
    negros saturados = pic<limite</pre>
    blancos saturados = pic>=limite
    pic[negros_saturados] = 0
    pic[blancos saturados] = 255
    return pic
Definimos el array "Zona Oscura"
zona oscura = np.full(input pic.shape,np.nan)
limite = np.nanmean(input pic)
for (coor_y,coor_x), value in np.ndenumerate(input_pic):
    aux = mask[coor y:coor y+1,coor x:coor x+1]
    if aux<limite: #Good Pixel</pre>
        zona oscura[coor y,coor x] = input pic[coor y,coor x]
    elif aux>limite: #Bad Pixel
        None
imprimir pic(zona oscura, "zona oscura")
            zona oscura
```



```
Definimos el array "Zona Clara"
zona_clara = np.full(input_pic.shape,np.nan)
limite = np.nanmean(input pic)
```

```
for (coor_y,coor_x), value in np.ndenumerate(input_pic):
    aux = mask[coor_y:coor_y+1,coor_x:coor_x+1]
    if aux>limite: #Good Pixel
        zona_clara[coor_y,coor_x] = input_pic[coor_y,coor_x]
    elif aux<=limite: #Bad Pixel
        None

imprimir pic(zona clara, "zona clara")</pre>
```

zona clara



Definimos la funcion que nos permite conocer el punto optimo para contrastar la imagen def contraste_optimo_por_MSE(segmento,pic):

```
output = 255 - np.nan_to_num(segmento,nan=0)

base = 255 -pic
errorlist = []
for i in range(30):

    f = funcion_de_contraste(output,i)
    error = np.sum((base-f)**2)
    errorlist.append(error)

m = np.arange(30)
#plt.scatter( m,errorlist)
#plt.title("Error vs intensidad (por percentile)")
```

```
#plt.show()
return (m[np.argmin(errorlist)]+1)
```

METODO BASE

```
Obtenemos el histograma de la zona clara con el fin de obtener su moda

a,b = np.histogram(zona_clara,bins=np.arange(0,270,10))

plt.bar( b[0:26],a[0:26],width=10)

moda zona clara = b[np.argmax(a)+1]
```

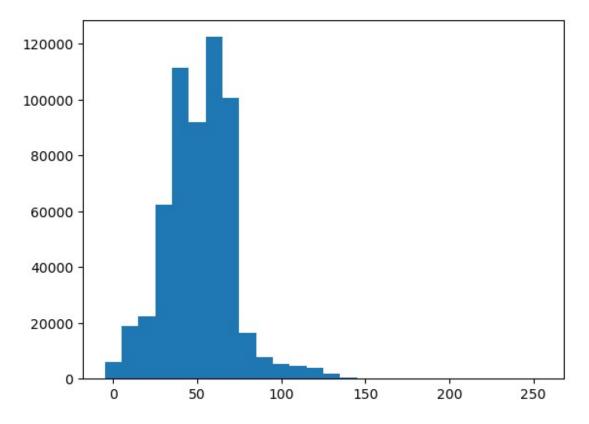
```
250000 - 200000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 150000 - 150000 - 150000 - 150000 - 150000 - 1500000 - 150000 - 1500000 - 1500000 - 1500000 - 1500000 - 1500000 - 1500000 - 1500000 - 15000000 - 1500000 - 1500000 - 1500000 - 1500000000 - 15000000 - 1500000 - 1500000 - 1500000 - 15000000000 - 1
```

```
Obtenemos el histograma de la zona oscura con el fin de obtener su moda

a,b = np.histogram(zona_oscura,bins=np.arange(0,270,10))

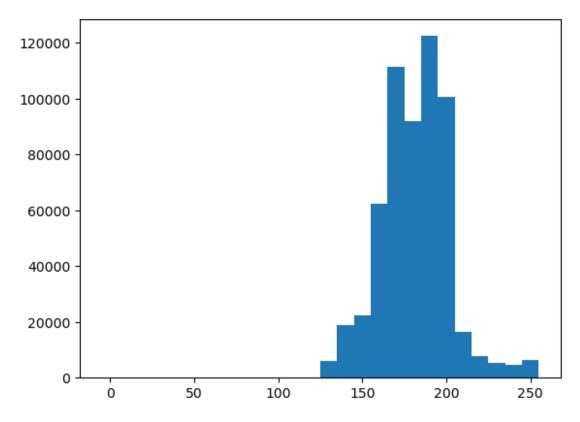
plt.bar(b[0:26],a[0:26],width=10)

moda_zona_oscura = b[np.argmax(a)+1]
```



```
Modificamos el histograma de la zona oscura con el fin de igualar la moda de ambas zonas osc = np.copy(zona_oscura) + ( -moda_zona_oscura + moda_zona_clara) #osc = zona_oscura lessThen0 = osc<0 moreThen255 = osc>255 osc[lessThen0] = 0 osc[moreThen255] = 255 #imprimir_pic(osc, "osc") a,b = np.histogram(osc,bins=np.arange(0,270,10)) plt.bar(b[0:26],a[0:26],width=10) #moda_zona_oscura = b[np.argmax(a)+1]
```

<BarContainer object of 26 artists>



imprimir_pic(osc,"Zona oscura con histograma modificado")

Zona oscura con histograma modificado



imprimir_pic(zona_clara,"zona_clara")

zona_clara



Combinamos ambas zonas: la zona clara y la zona oscura modificada

```
z1 = np.nan_to_num(zona_clara,0)
z2 = np.nan_to_num(osc,0)

suma_parcial = (z1 + z2)
imprimir_pic(suma_parcial, "combianadas")
```

combianadas



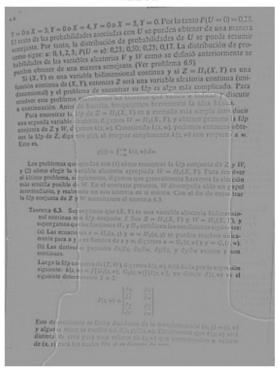
Restamos las mascara (en tonos de gris) de la imagen original

```
resta_parcial = np.copy(suma_parcial)
mask_gris = ndimage.median_filter(suma_parcial,
size=15,mode='reflect')

resta_parcial = ((suma_parcial - mask_gris)+255)/(2)
lessThen0 = resta_parcial<0
moreThen255 = resta_parcial>255
resta_parcial[lessThen0] = 0
resta_parcial[moreThen255] = 255

imprimir_pic(resta_parcial, "resta")
```

resta



Aplicamos un contraste optmizado por MSE a la imagen de "resta parcial"

```
p_indicado = contraste_optimo_por_MSE(255 - resta_parcial,255 -
input_pic)
final_metodo_BASE = funcion_de_contraste(resta_parcial,p_indicado)
imprimir_pic(final_metodo_BASE,"final_metodo_BASE")
```

final metodo BASE

```
y = 0 o X = 3, Y = 0 o X = 4, Y = 0 o X = 5, Y = 0. For lo tanto P(U = 0) = 0.23.

y = 0 o X = 3, Y = 0 o X = 4, Y = 0 o X = 5, Y = 0. For lo tanto P(U = 0) = 0.23.

y = 0 o X = 3, Y = 0 o X = 4, Y = 0 o X = 5, Y = 0. For lo tanto P(U = 0) = 0.23.

y = 0 o X = 3, Y = 0 o X = 4, Y = 0 o X = 5, Y = 0. For lo tanto P(U = 0) = 0.23.

It can de las probabilidades aucciadas con U as pueden obtener de una manera semigianto. (Ver, problemá 69).

y = 0 o X = 3, Y = 0 o X = 4, Y = 0. O X = 0. O X = 0. O X = 0.

y = 0 o X = 1, Y = 0 o X = 0. O X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 1, Y = 0 o X = 0.

y = 0 o X = 0 o X = 0.

y = 0 o X = 0
```

METODO B

```
# Importar las librearias necesarias:
```

```
import matplotlib.pyplot as plt
from PIL import Image
import scipy.ndimage as snd
import numpy as np
import cv2
from skimage.filters import threshold otsu
from cv2 import threshold, adaptiveThreshold
# Declarar metodos para aplicar Clausura y Threshold:
def closing(image array):
  # Estructurar elemento de clausura:
  huella = np.ones((40, 40))
  # Aplicar clausura de grises:
  fondo = snd.grey_closing(image_array, footprint=huella)
  # Se resta el fondo de la imagen:
  fondo libre = (image array.astype(np.float64) -
fondo.astype(np.float64))
  # Se reescala la imagen con el fondo libre de 0 a 255:
  denominador = (fondo libre.max() - fondo libre.min())
```

```
fondo libre normalizado = (fondo libre - fondo libre.min())*
255/denominador
  # Convertir fondo libre normalizado a uint8:
  fondo libre normalizado = fondo_libre_normalizado.astype(np.uint8)
  # Convertir fondo libre normalizado a imagen:
  fondo libre normalizado = Image.fromarray(fondo libre normalizado)
  return fondo libre normalizado
def threshold(image array):
  th = cv2.adaptiveThreshold(np.asarray(image_array),
    255, # Maximo valor assignado al valor de un pixel que excede el
'threshold'.
    cv2.ADAPTIVE THRESH MEAN C, # suma ponderada "gaussiana" de los
vecinos.
    cv2.THRESH BINARY, # Tipo de threshold.
    15, # Tamaño del bloque (ventana 5x5)
    12)
  # Convertir th a imagen:
  th = Image.fromarray(th)
  return th
# Importar imagen:
input image = input pic
# Convertir la imagen en arreglo:
image array = np.asarray(input image)
# Aplicar Clausura:
closing image = closing(image array)
# Aplicar Threshold:
threshold image = threshold(closing image)
input image
array([[146, 148, 151, ..., 179, 179, 178],
       [141, 143, 146, \ldots, 179, 178, 178],
       [139, 141, 144, ..., 178, 178, 177],
       [190, 190, 190, ...,
                             36,
                                  36,
                                       36],
       [190, 190, 190, ...,
                             36,
                                  36,
                                       361,
       [190, 190, 190, ...,
                             36,
                                  36,
                                       36]], dtype=uint8)
closing image
```

 $Y = 0 \circ X = 3$, $Y = 0 \circ X = 4$, $Y = 0 \circ X = 5$, Y = 0. Por lo tanto P(U = 0) = 0.28. El resto de las probabilidades asociadas con U se pueden obtener de una manera semejante. Por tanto, la distribución de probabilidades de U se puede resumir semejane: u: 0, 1, 2, 3; P(U = u): 0.28; 0.30; 0.25; 0.17. La distribución de procomo sigue: u: 0, 1, 2, 3; P(U = u): 0.28; 0.30; 0.25; 0.17. babilidades de las variables aleatorias V y W como se definió anteriormente se pueden obtener de una manera semejante. (Ver problema 6.9).

Si (X, Y) es una variable bidimensional continua y si $Z = H_1(X, Y)$ es una función continua de (X, Y), entonces Z será una variable aleatoria continua (unidimensional) y el problema de encontrar su fdp es algo más complicado. Para resolver este problema necesitamos un teorema que vamos a indicar y discutir a continuación. Antes de hacerlo, bosquejemos brevemente la idea básica.

Para encontrar la fdp de $Z = H_1(X, Y)$ es a menudo más simple introducir una segunda variable aleatoria, digamos $W = H_2(X, Y)$, y obtener primero la fdp conjunta de Z y W, digamos k(z, w). Conociendo k(z, w), podemos entonces obtener la sdp de Z, digamos g(z), al integrar simplemente k(z, w) con respecto a w.

$$g(z) = \int_{-\infty}^{+\infty} k(z, w) dw.$$

Los problemas que quedan son (1) cómo encontrar la fdp conjunta de Z y W, y (2) cómo elegir la variable aleatoria apropiada $W = H_2(X, Y)$. Para resolver el último problema, simplemente, digamos que generalmente hacemos la elección más sencilla posible de W. En el contexto presente, W desempeña sólo un papel intermediario, y realmente no nos interesa en sí misma. Con el fin de encontrar la sdp conjunta de Z y W necesitamos el teorema 6.3.

Teorema 6.3. Supongamos que (X, Y) es una variable aleatoria bidimensional continua con fdp conjunta f. Sea $Z = H_1(X, Y)$ y $W = H_2(X, Y)$, y supongamos que las funciones H_1 y H_2 satisfacen las condiciones siguientes: (a) Las ecuaciones $z = H_1(x, y)$ y $w = H_2(x, y)$ se pueden resolver únicamente para x y y en función de z y w, digamos $x = G_1(z, w)$ y $y = G_2(z, w)$. (b) Las derivadas parciales $\partial x/\partial z$, $\partial x/\partial w$, $\partial y/\partial z$, y $\partial y/\partial w$ existen y son

Luego la fdp conjunta de (Z, W), digamos k(z, w), está dada por la expresión signiente: $k(z, w) = f[G_1(z, w), G_2(z, w)] |J(z, w)|$, en donde J(z, w) es el siguiente determinante 2 x 2:

$$J(z, w) = \begin{vmatrix} \frac{\partial x}{\partial z} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial z} & \frac{\partial y}{\partial w} \end{vmatrix}$$

Este determinante se llama Jacobiano de la transformación $(x, y) \rightarrow (z, w)$ y algunas veces se escribe así $\partial(x, y)/\partial(z, w)$. Observemos que k(z, w) será distinta de cero para esos valores de (z, w) que corresponden a valores de (x, y) para los cuales f(x v) es distinto da

Y = 0 o X = 3, Y = 0 o X = 4, Y = 0 o X = 5, Y = 0. Por lo tanto P(U = 0) = 0.28. El resto de las probabilidades asociadas con U se pueden obtener de una manera semejante. Por tanto, la distribución de probabilidades de U se puede resumir semejante. Por tanto, la distribución de probabilidades de las variables aleatorias V y W como se definió anteriormente se babilidades de las variables aleatorias V y W como se definió anteriormente se pueden obtener de una manera semejante. (Ver problema 6.9).

Si (X, Y) es una variable bidimensional continua y si $Z = H_1(X, Y)$ es una función continua de (X, Y), entonces Z será una variable aleatoria continua (unidimensional) y el problema de encontrar su fdp es algo más complicado. Para resolver este problema necesitamos un teorema que vamos a indicar y discutir a continuación. Antes de hacerlo, bosquejemos brevemente la idea básica.

Para encontrar la sdp de $Z = H_1(X, Y)$ es a menudo más simple introducir una segunda variable aleatoria, digamos $W = H_2(X, Y)$, y obtener primero la sdp conjunta de Z y W, digamos k(z, w). Conociendo k(z, w), podemos entonces obtener la sdp de Z, digamos g(z), al integrar simplemente k(z, w) con respecto a w. Esto es.

$$g(z) = \int_{-\infty}^{+\infty} k(z, w) dw.$$

Los problemas que quedan son (1) cómo encontrar la fdp conjunta de Z y W, y (2) cómo elegir la variable aleatoria apropiada $W = H_2(X, Y)$. Para resolver el último problema, simplemente, digamos que generalmente hacemos la elección más sencilla posible de W. En el contexto presente, W desempeña sólo un papel intermediario, y realmente no nos interesa en si misma. Con el fin de encontrar la fdp conjunta de Z y W necesitamos el teorema 6.3.

Teorema 6.3. Supongamos que (X, Y) es una variable aleatoria bidimensional continua con fdp conjunta f. Sea $Z = H_1(X, Y)$ y $W = H_2(X, Y)$, y supongamos que las funciones H_1 y H_2 satisfacen las condiciones siguientes:

(a) Las ecuaciones $z = H_1(x, y)$ y $w = H_2(x, y)$ se pueden resolver unicamente para x y y en función de z y w, digamos $x = G_1(z, w)$ y $y = G_2(z, w)$.

(b) Las derivadas parciales $\partial x/\partial z$, $\partial x/\partial w$, $\partial y/\partial z$, y $\partial y/\partial w$ existen y son continuas.

Luego la fdp conjunta de (Z, W), digamos k(z, w), está dada por la expresión siguiente: $k(z, w) = f[G_1(z, w), G_2(z, w)] |J(z, w)|$, en donde J(z, w) es el siguiente determinante 2×2 :

$$J(z, w) = \begin{vmatrix} \frac{\partial x}{\partial z} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial z} & \frac{\partial y}{\partial w} \end{vmatrix}$$

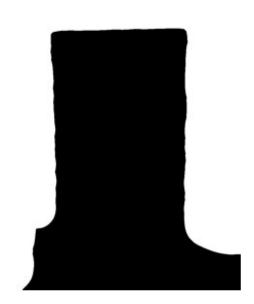
Este determinante se llama Jacobiano de la transformación $(x, y) \rightarrow (z, w)$ y algunas veces se escribe así $\partial(x, y)/\partial(z, w)$. Observemos que k(z, w) será distinta de cero para esos valores de (z, w) que corresponden a valores de (x, y) para los cuales f(x, y) es distinte de

```
imprimir_pic(mask,'Máscara')
imprimir_pic(input_pic,'original')
```

/tmp/ipykernel_12094/2346952556.py:1: FutureWarning: `selem` is a
deprecated argument name for `dilation`. It will be removed in version
1.0. Please use `footprint` instead.

mask = dilation(image=mask, selem=disk(40))

Máscara



original

```
y=00 X=3, Y=00 X=4, Y=00 X=5, Y=0. Por lo tanto P\{U=0\}=0.28. It can de las probabilidades asociadas con U se pueden obtener de una manera El caño de las probabilidades asociadas con U se pueden obtener de una manera como sigas: u^01, u^12, u^12, u^13, u^24, u^24, u^25, u^27, u^27, u^27, u^28, u^28, u^29, u^2
```

```
zona_oscura = np.full(input_pic.shape,np.nan)
limite = np.nanmean(input pic)
for (coor_y,coor_x), value in np.ndenumerate(input_pic):
    aux = mask[coor_y:coor_y+1,coor_x:coor_x+1]
if aux<limite: #Good Pixel</pre>
         zona_oscura[coor_y,coor_x] = input_pic[coor_y,coor_x]
    elif aux>limite: #Bad Pixel
         None
imprimir pic(zona oscura, "zona oscura")
```

zona oscura

```
de Z=H_1(X,Y) es a menudo más simple à oria, digamos W=H_1(X,Y), y obtener prime qui k(z,w). Conociendo k(z,w), podemos entos \psi(z), al integrar simplemente h(z,w) con resp \psi(z), al integrar simplemente h(z,w) con resp \psi(z)=\int_{-\infty}^{\infty}k(z,w)\,dw.

dan son (1) como encontrar la folp conjunta d' yle aleatoria apropiada W=H_1(X,Y). Paraz mente, diganno que generalmente hacemos la En el contesto presente, W desempeña sólo s no son interesa en si misma. Con el fin de el vecesitamos el teorema 6.3.

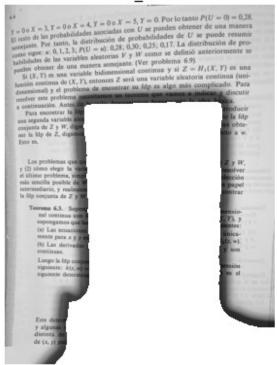
Amos que (X,Y) en una variable aleatoria bid que conjunta f. Sea Z=H_1(X,Y) y W=H_2(X,Y) conjunta f. Sea Z=H_2(X,Y) y W=H_2(X,Y) conjunta f. Sea Z=H_2(X,Y) y W=H_2(X,Y) in Y=1 d'uniciones H_1(X,Y) y Y=H_2(X,Y) pueden resolve innoiso de z y, y, digamos x=G_1(x,Y) y y=1 arciales d_{X}(z), d_{X
```

```
zona_clara = np.full(input_pic.shape,np.nan)
limite = np.nanmean(input_pic)

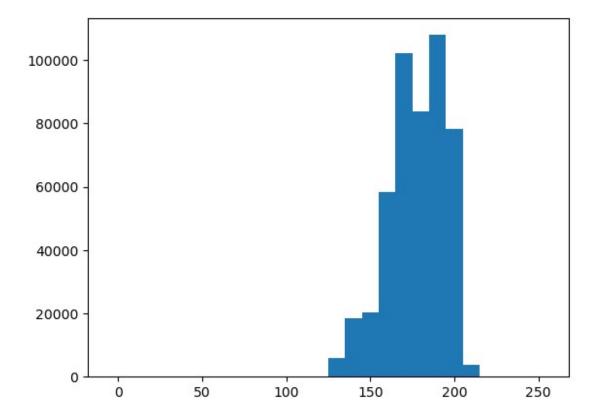
for (coor_y,coor_x), value in np.ndenumerate(input_pic):
    aux = mask[coor_y:coor_y+1,coor_x:coor_x+1]
    if aux>limite: #Good Pixel
        zona_clara[coor_y,coor_x] = input_pic[coor_y,coor_x]
    elif aux<=limite: #Bad Pixel
        None

imprimir_pic(zona_clara,"zona_clara")</pre>
```

zona_clara



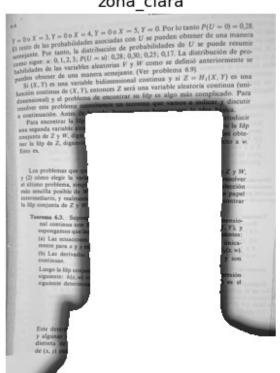
```
\#a,b = np.histogram(zona clara,bins=np.arange(0,270,10))
\#plt.bar(b[0:26],a[0:26],width=10)
\# moda\ zona\ clara = b[np.argmax(a)+1]
\#a,b = np.histogram(zona oscura,bins=np.arange(0,270,10))
#plt.bar( b[0:26],a[0:26],width=10)
\#moda\ zona\ oscura = b[np.argmax(a)+1]
osc = np.copy(zona oscura) + ( -moda zona oscura + moda zona clara)
#osc = zona oscura
lessThen0 = osc<0
moreThen255 = osc>255
osc[lessThen0] = 0
osc[moreThen255] = 255
#imprimir pic(osc, "osc")
a,b = np.\overline{h}istogram(osc,bins=np.arange(0,270,10))
plt.bar(b[0:26],a[0:26],width=10)
\# moda\ zona\ oscura = b[np.argmax(a)+1]
imprimir pic(osc, "Zona oscura con histograma modificado")
imprimir pic(zona clara, "zona clara")
```



Zona oscura con histograma modificado



zona_clara



```
z1 = np.nan_to_num(zona_clara,0)
z2 = np.nan_to_num(osc,0)
suma_parcial = (z1 + z2)
imprimir_pic(suma_parcial,"combianadas")
```

combianadas

```
y = 0 0 X = 3, Y = 0 0 X = 4, Y = 0 0 X = 5, Y = 0. Por lo tanto P{U = 0} = 0,28, Y = 0 0 X = 3, Y = 0 0 X = 4, Y = 0 0 X = 5, Y = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0,28, D = 0. Por lo tanto P{U = 0} = 0. Por lo tanto P{U = 0} = 0. Por lo tanto P{U = 0} = 0. Por lo tanto P{U
```

```
resta_parcial = np.copy(suma_parcial)
mask_gris = ndimage.median_filter(suma_parcial,
size=15,mode='reflect')

resta_parcial = ((suma_parcial - mask_gris)+255)/(2)
lessThen0 = resta_parcial<0
moreThen255 = resta_parcial>255
resta_parcial[lessThen0] = 0
resta_parcial[moreThen255] = 255

imprimir_pic(resta_parcial, "resta")
p_indicado = contraste_optimo_por_MSE(255 - resta_parcial,255 - input_pic)
final_metodo_A = funcion_de_contraste(resta_parcial,p_indicado)
imprimir_pic(final_metodo_A, "final_metodo_A")
```

y=00 X=3, Y=00 X=4, Y=00 X=5, Y=0. Per lotatato F(U=0)=0.22. It can do has probabilidades accetadas con U as pureden obtener de una manera di resta de has probabilidades de U se puede resumis integrats. Per tanto, la distribución de probabilidades de U se puede resumis integrats. Per tanto, la distribución de y V9 V1 como se definido anteriormente se habilidades de las variables aleatorias V9 V1 como se definido anteriormente se pueden obtener de una nariable bidantencional continua y si $Z=H_1(X,Y)$ es una variable bidantencional continua y si $Z=H_1(X,Y)$ es una habilida continua de (X,Y)1 contones Z1 será una variable aleatorias continua (unidentational) y de problema de encontrar su fido es algo más complicado. Para recolver este problema exclusivas una transma que valunca a indicar y discutir a continuação. Anter el Asserba, hosopraphos becomens la lafea V1 de este esta problema de V2 será de V3 en en mendo en las simple introducir esta esta partida variable elas rosa, disprante V3 V4 V4, y contrar partida de V5 V6, de V6 contrar la V7 V8 en en mendo en la simple introducir esta esta partida variable elas rosa, disprante V3 V4 V6, y contrar partida de V5 V6, de probabilita de V6 V7, de V7, de V8 V8, de probabilita de V8 V8, de probabilita de V9 V9. Con respecto a V9, de probabilita de V9 V9. Con respecto a V9. Con respecto a V9.

final_metodo A

$$g(z) = \int_{-\infty}^{+\infty} k(z, w) dw.$$

Les problemas que que dan son (1) cômo encontrar la filp conjunta de Z y W, y (2) cêmo elegir la variable abenteni apropiada $W = H_1(X,Y)$. Para proolver el étimos problema, simplement, digamos que generalmente hacemos la elección más senella poible de W. En el contexto presente, W desempela ablo un papel : (- internedirar) y rendroste no son interesa en el misma. Con el fin de escontrar : la filp conjunta de Z y W inconitamos el teorema 6.3.

stream 6.5. Supposymmos que (X,Y) es uns variable alcatoria bid mensional soutinus con tip conjunta f, Sen $Z=H_1(X,Y)$ y $W=H_2(X,Y)$, f, appengames que las funciones H_1 y H_2 authorizante a conditioner significante H_1 y H_2 and H_2 and H_3 y H_3 and H_4 in the funcione H_3 H_4 and H_4 is a puncion resolver unication funcion for H_4 y H_4 and H_4 is a puncion H_4 y H_4 and H_4 is a puncion H_4 y H_4 and H_4 is a funcion of H_4 in H_4 and H_4 is a funcion of H_4 in H_4

Lacgo la lép conjunt a de (Z,W), digamos k(t,w), està dada por la sipresido signiente: $k(t,w) = f(G_1(t,w), G_2(t,w)] [J(t,w)]$, en donde J(t,w) es et (t,w) es (t,w) es (t,w).

$$J(z,w) = \begin{vmatrix} \frac{\partial x}{\partial z} & \frac{\partial z}{\partial w} \\ \frac{\partial y}{\partial z} & \frac{\partial y}{\partial z} \end{vmatrix}$$

Esta determinante se llama Jacobieno de la transformación $(x,y) \rightarrow (x,w)$ y alguna veces se service sal $\partial(x,y)\partial(t,w)$. Observemos que $\lambda(x,y)$ será distinta de jarro para estos valores de (x,y) que corresponden a valores de (x,y) para les cuales f(x,y) actá distinta de (x,y) que corresponden a valores de (x,y) actá de (x,

```
plt.imshow(input pic, cmap = 'gray')
plt.title('original')
plt.axis('off')
plt.subplot(1,4, 2)
plt.imshow(final metodo BASE, cmap = 'gray')
plt.axis('off')
plt.title('metodo BASE')
plt.subplot(1,4, 3)
plt.imshow(final metodo A, cmap = 'gray')
plt.axis('off')
plt.title('metodo A')
plt.subplot(1,4, 4)
plt.imshow(threshold image, cmap = 'gray')
plt.axis('off')
plt title('metodo B')
plt.show()
       original
                       metodo BASE
                                                           metodo B
                                         metodo A
```

Guardamos el mejor resultado como archivo jpeg type (threshold image)

```
PIL.Image.Image
nombre = imagen_entrada + "_PROCESADA.jpeg"
threshold image.save(nombre)
```