Pr?ctica 13

May 20, 2019

1 Práctica 13: Análisis de imágenes.

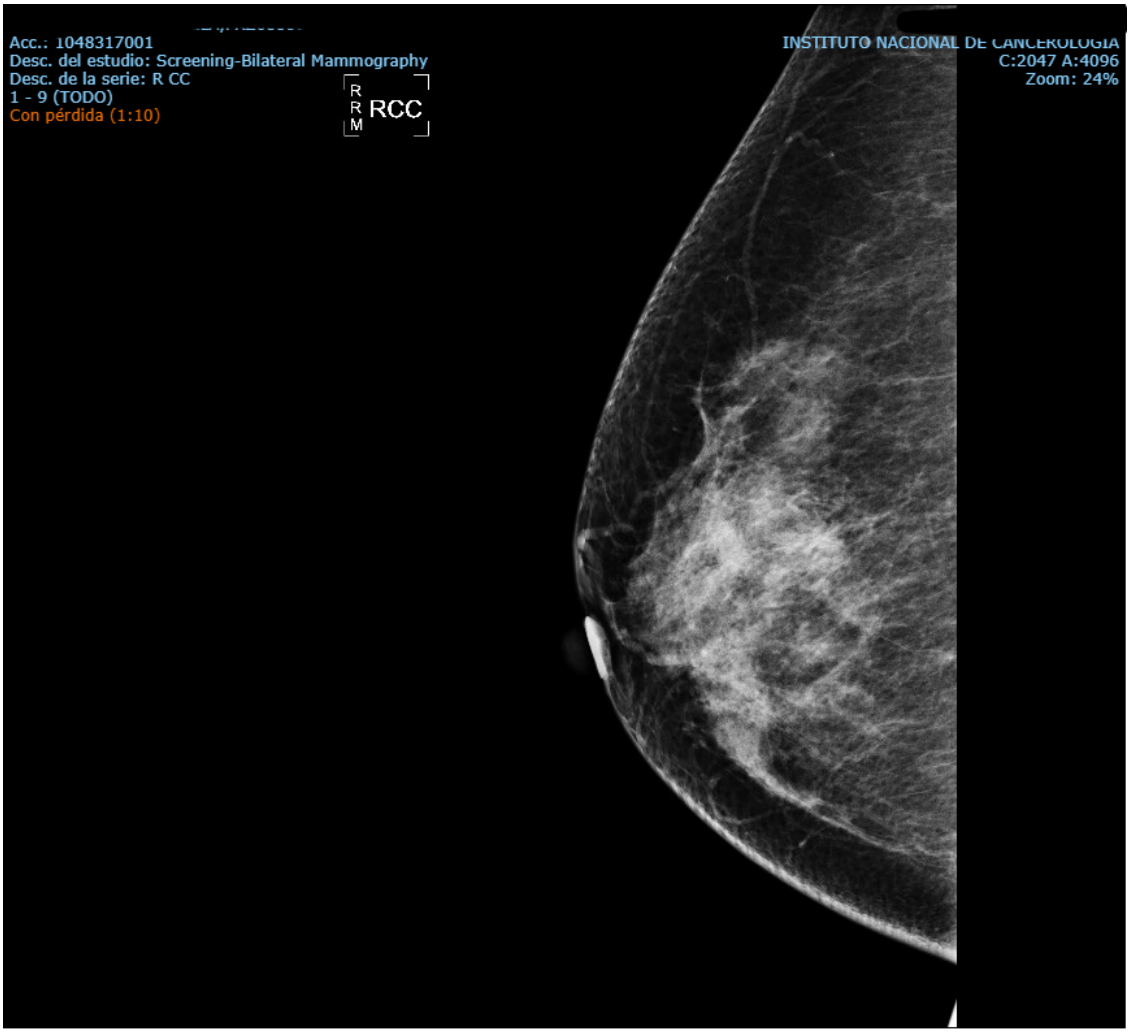
Al iniciar el trabajo en estas prácticas se tomaron como datos de entrada un csv con el título de "Breast Cancer Winsconsin (Diagnostic) Data Set" que contiene las características de una aspiración de aguja de una anomalía encontrada en una mamografía. Estos datos se utilizaron con la finalidad de complementar los conocimientos sobre el cáncer de mama y las posibles herramientas de diagnóstico que pueden crearse.

En esta práctica se quieren tomar las herramientas de manipulación de imágenes tales como OpenCV, ImageMagick, y Pillow, todas ellas controlables en base a Python o la terminal del ordenador, y modificar una imagen de mamografía para en este caso poder realizar lo que se conoce como máscara ROI, que es básicamente un parche que utilizan los médicos para localizar las anomalías dentro de las imágenes.

Para iniciar se importa la imagen de prueba, y las librerías que se van a utilizar. Ya que esta base de datos aún está en proceso de ser publica, aún tiene marcados los datos en sus esquinas. Se borró a mano el nombre del paciente y los datos de fecha de publicación, siguiendo la normativa programada por el Instituto Nacional de Cancerología para proteger los datos de los pacientes.

```
In [8]: from PIL import Image, ImageFilter, ImageEnhance
        i = Image.open("maligna-1.png")
        i
```

```
Out[8]:
```



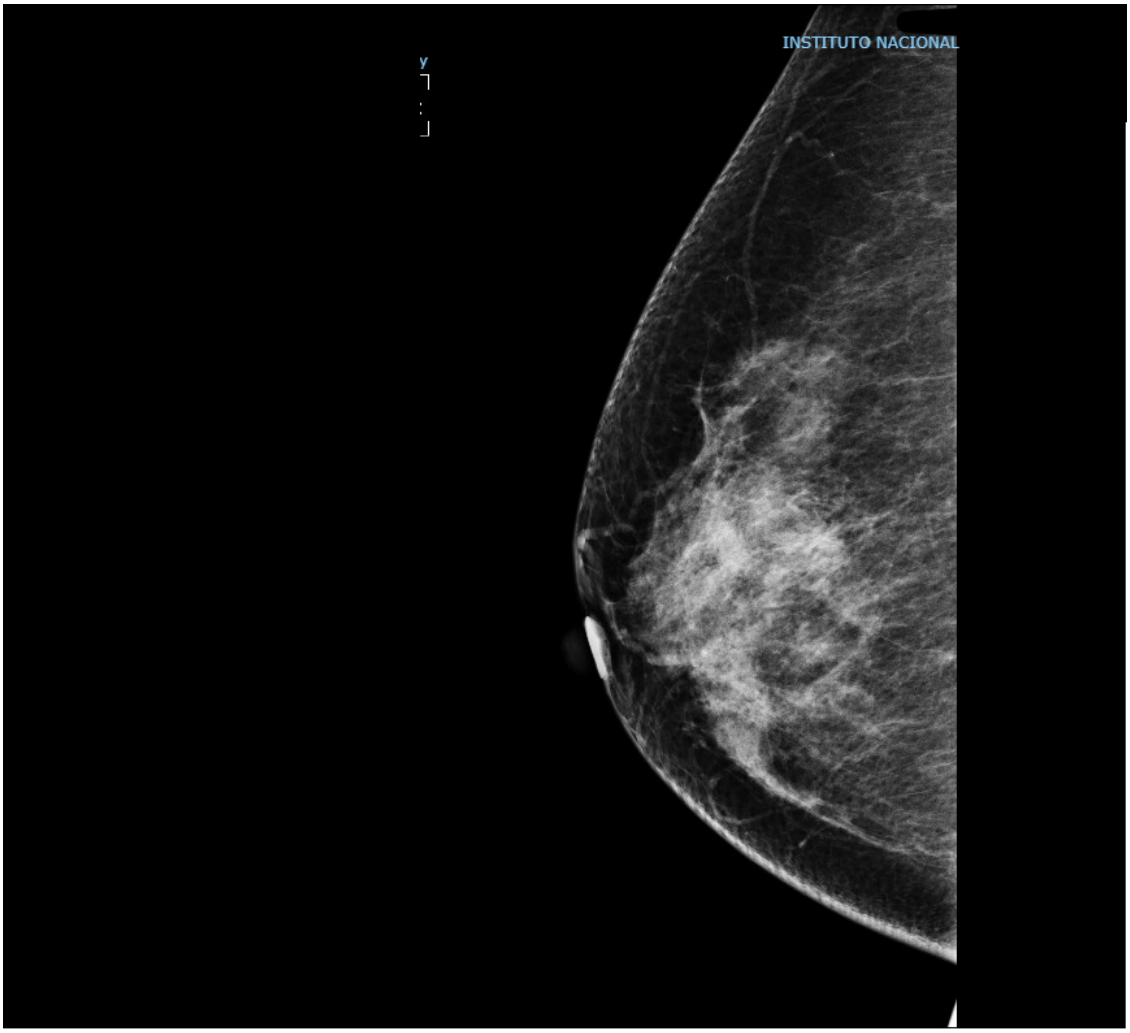
A pesar de que ya no está el nombre del paciente, aún causa ruido todo el demás texto, por lo que se utilizan parches negros para tapar los datos creados con la herramienta PIL. Una vez creados, se guarda la imagen nueva para poder seguir manipulandola.

```
In [9]: from PIL import Image, ImageChops, ImageFont, ImageDraw, ImageEnhance
        import numpy as np

        draw = ImageDraw.Draw(i)
        #draw.rectangle(((0, 0), (0, 0)), fill="black")
        draw.rectangle(((0, 0), (320, 110)), fill="black")
        draw.rectangle(((740, 0), (900, 90)), fill="black")

        i.save("crop-mal1.png")
        crop = Image.open("crop-mal1.png")
        crop

Out[9]:
```



Teniendo la imagen de esta manera, se pueden recortar los bordes negros, para que la imagen pueda ser más pequeña y ocupe menos procesamiento. El recorte se realiza fuera del 'notebook' dentro de la terminal del sistema. El resultado es el de la imagen llamada 'image_out.png'.

Después de realizar este recorte, se le da a la imagen una serie de filtros, todos documentados en la página de la librería PIL.

Los filtros utilizados son 'Sharp', 'Detail' y 'Contour'. Este último se desprecia ya que lo que se quiere lograr es un detalle de las posibles anomalías, y este filtro le quita muchas de las características importantes para ayudar esta tarea.

```
In [6]: i = Image.open("image_out.png")
        sharpener = ImageEnhance.Sharpness (i)
        sharpened = sharpener.enhance(3.0)
        sharpened.save('Sharp-mal1.png')
        ima2 = Image.open("Sharp-mal1.png")
        #im.save('normales/no-{}.png'.format(i))
        ima2
```

Out[6]:


17/12/2
INSTITUTO NACIONAL

In [32]:
```
from PIL import Image, ImageFilter, ImageEnhance
i = Image.open("Sharp-mal1.png")

i.filter(ImageFilter.DETAIL).save("detail-m1.png")
detail = Image.open("detail-m1.png")
#im.save('normales/no-{}.png'.format(i))
detail
```

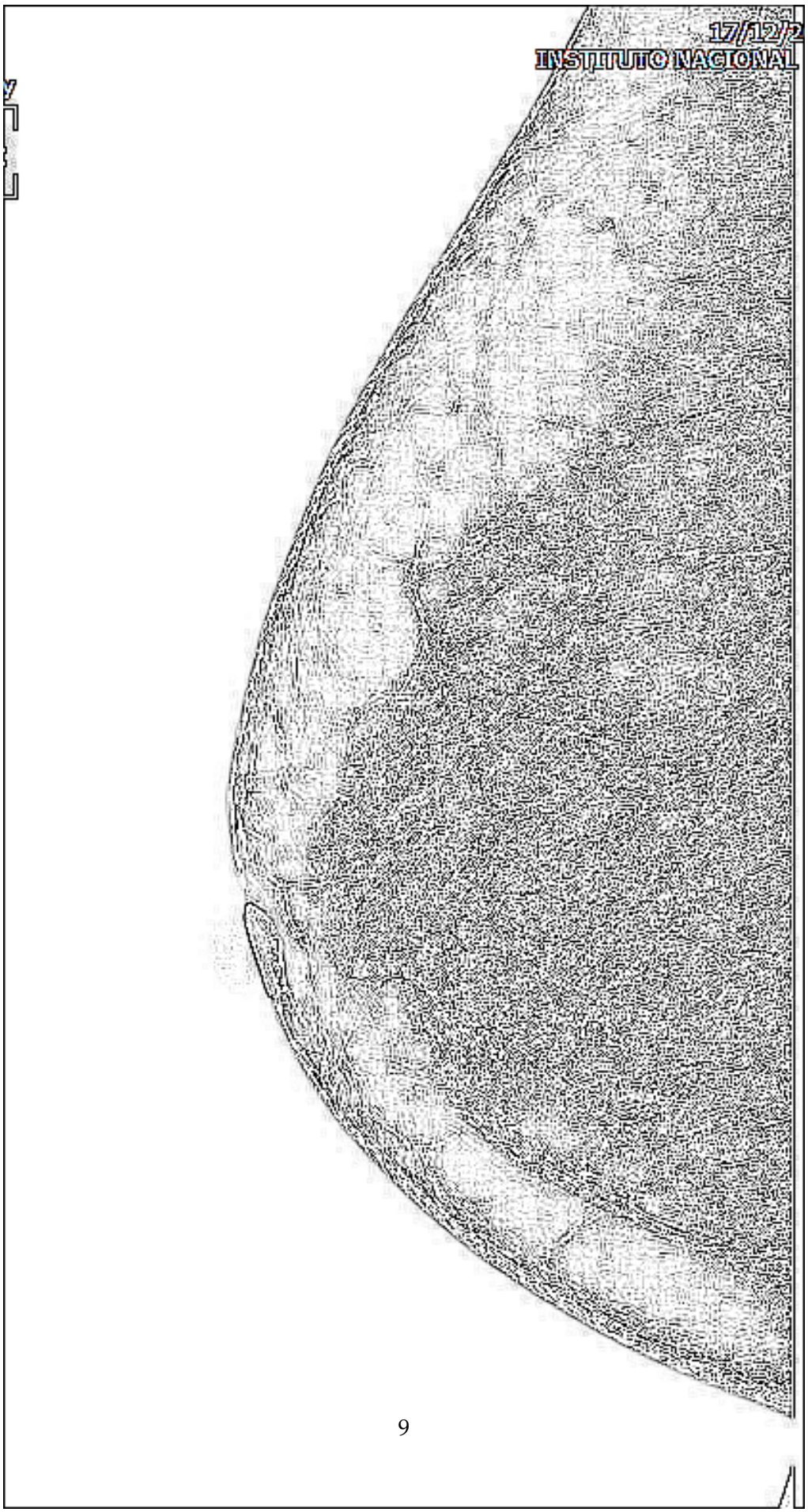
Out[32]:


17/12/2
INSTITUTO NACIONAL

In [33]:
```
from PIL import Image, ImageFilter, ImageEnhance
i = Image.open("detail-m1.png")

i.filter(ImageFilter.CONTOUR).save("contour-m1.png")
contour = Image.open("contour-m1.png")
contour
```

Out[33]:

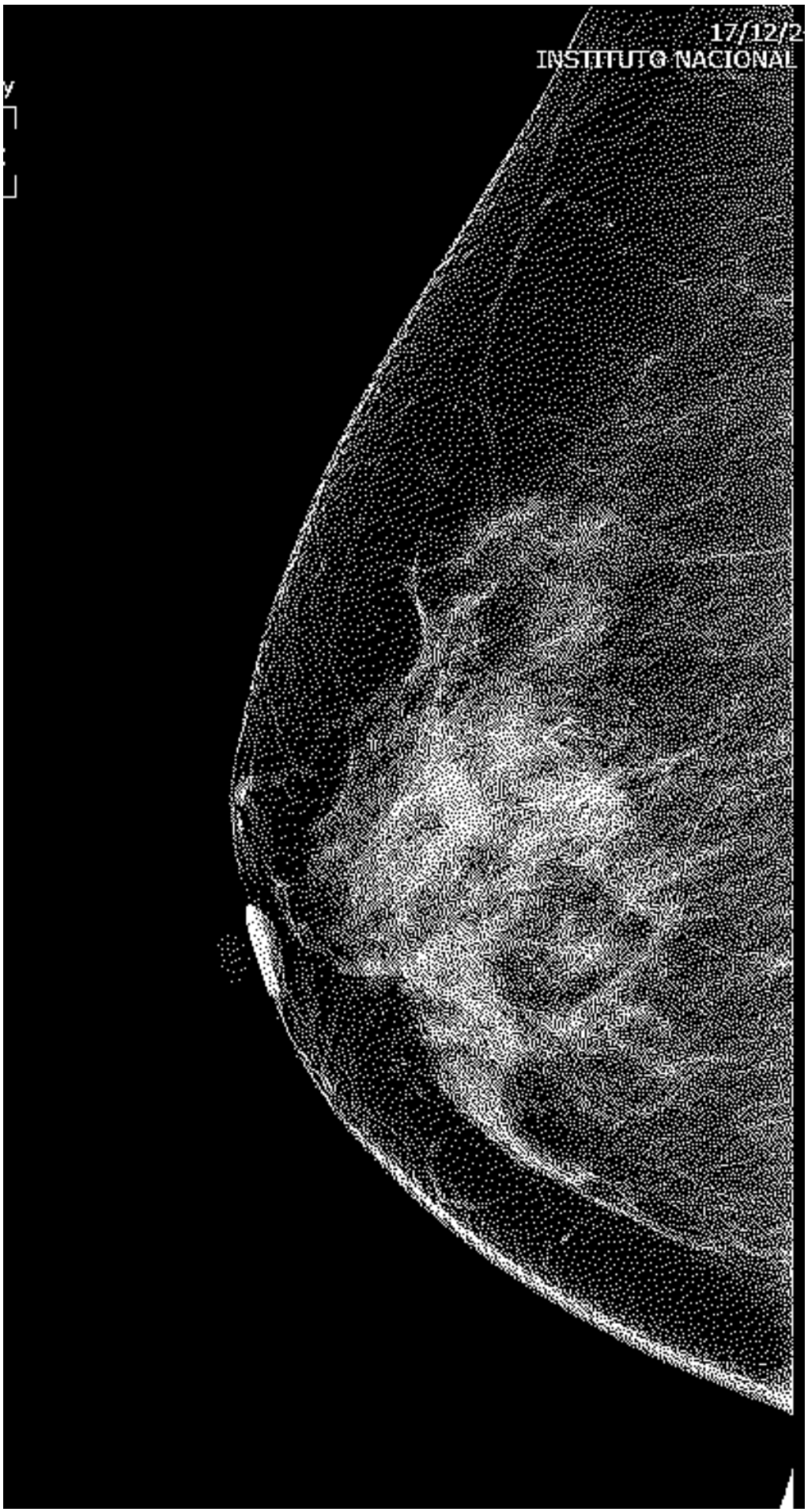

17/12/2
INSTITUTO NACIONAL

Las imágenes de mamografía más antiguas están en formato de blanco y negro, como es el ejemplo de la base de datos Mini-Mias, que ha estado disponible desde los inicios del año 1999. Bases de datos más recientes, como lo es la base de la que se tomó la imagen de ejemplo para esta práctica ya están desarrolladas en escala RGB para poder mejorar su calidad. Como paso siguiente en esta práctica se transforma la imagen a blanco y negro, siguiendo los pasos de la práctica de prueba de la Dra. Elisa.

```
In [34]: import ssl
         import requests
         imagen = Image.open("detail-m1.png")
         nuevo = imagen.convert('1')
         print(nuevo.size)
         nuevo
```

```
(419, 786)
```

Out[34]:


17/12/2
INSTITUTO NACIONAL

Con la imagen de esta manera se puede apreciar un poco más donde se encuentra la mayor concentración de masa. No es la norma, pero usualmente cuando existe una gran concentración de masa en la imagen es probable que ahi se encuentre una anomalía. Dependiendo de la forma de dicha masa es cuando los expertos la pueden clasificar en anomalía benigna o maligna.

El siguiente paso será el de limpiar esos pequeños puntos alrededor de la masa principal para poder crear la máscara final. En el ejemplo de la práctica, la Dra. Elisa establece que el umbral tomó varios intentos antes de llegar a uno significativo, por lo que se decidió también mover los datos. Entre menor el umbral, menos puntos aparecen, pero después de probar con números desde 50, y subiendo en intervalos de 10, se llegó a la conclusión de que el 150 es un umbral bueno para quitar los puntos, sin perder muchas características de la masa principal.

In [38]:
```
import ssl
import requests
from PIL import Image, ImageDraw
if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

negro = (0, 0, 0)
blanco = (255, 255, 255)
umbral = 150 #90 era el primero
n = Image.open("detail-m1.png")
w, h = n.size
P = n.load()
for f in range(h): # bordes verticales
    if P[0, f] == blanco:
        ImageDraw.floodfill(n, (0, f), negro)
    if P[w - 1, f] == blanco:
        ImageDraw.floodfill(n, (w - 1, f), negro)
for c in range(w): # bordes horizontales
    if P[c, 0] == blanco:
        ImageDraw.floodfill(n, (c, 0), negro)
    if P[c, h - 1] == blanco:
        ImageDraw.floodfill(n, (c, h - 1), negro)
for f in range(h):
    for c in range(w):
        rgb = P[c, f]
        if max(rgb) - min(rgb) > umbral: # tiene un color que no es gris
            P[c, f] = negro
b = n.convert('1') # binaricemos lo que queda
b.save('rgb-m1.png')
b
```

Out[38]:


17/12/2
INSTITUTO NACIONAL

Se tiene la imagen más limpia, pero aún puede hacerse una modificación para quitar esa parte de poca densidad en la parte derecha de la imagen. En el código anterior se quitaron los pixeles blancos que no tenían vecinos, para poder quitar esos puntos solitarios alrededor de la masa. En este siguiente código se quitan los puntos sin vecino, y con un solo vecino.

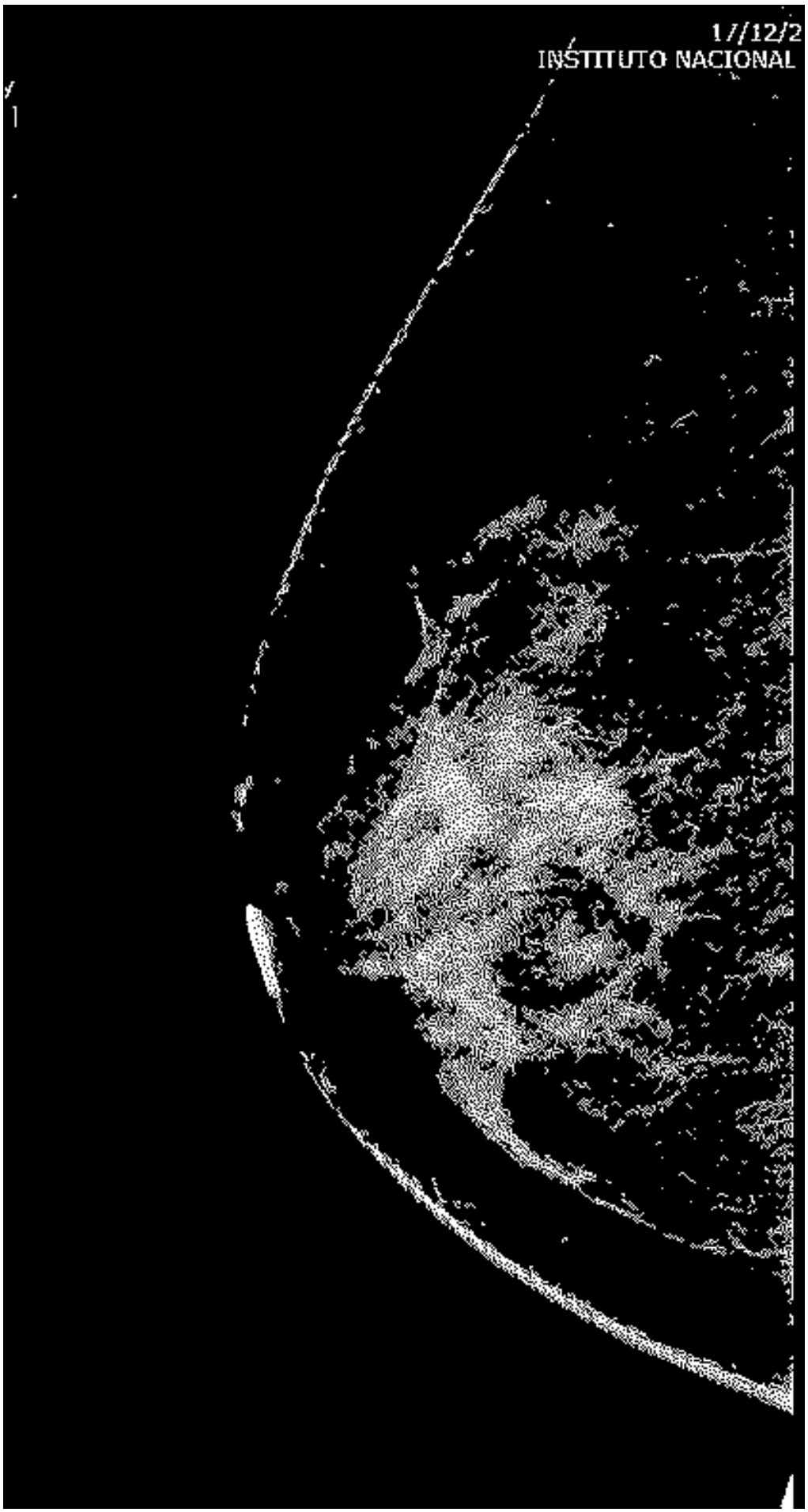
In [23]:
```
import ssl
import requests
from PIL import Image

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

nuevo = b
P = nuevo.load()
ancho, altura = nuevo.size
borrados = 0
vecinos = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
for fila in range(altura):
    for columna in range(ancho):
        if P[columna, fila] == 255: # pixel es negro
            contador = 0
            for (df, dc) in vecinos:
                vf = fila + df
                vc = columna + dc
                if vf >= 0 and vc >= 0 and vf < altura and vc < ancho: # si existe el
                    if P[vc, vf] == 255:
                        contador += 1
            if contador < 2: # uno o cero vecinos negros
                P[columna, fila] = 0 # será blanco
                borrados += 1
print(borrados, "pixeles negros eliminados")
nuevo.save('pixeles-m1.png')
pixeles = Image.open("pixeles-m1.png")
pixeles
```

```
13121 pixeles negros eliminados
```

Out[23]:


1//12/2
INSTITUTO NACIONAL

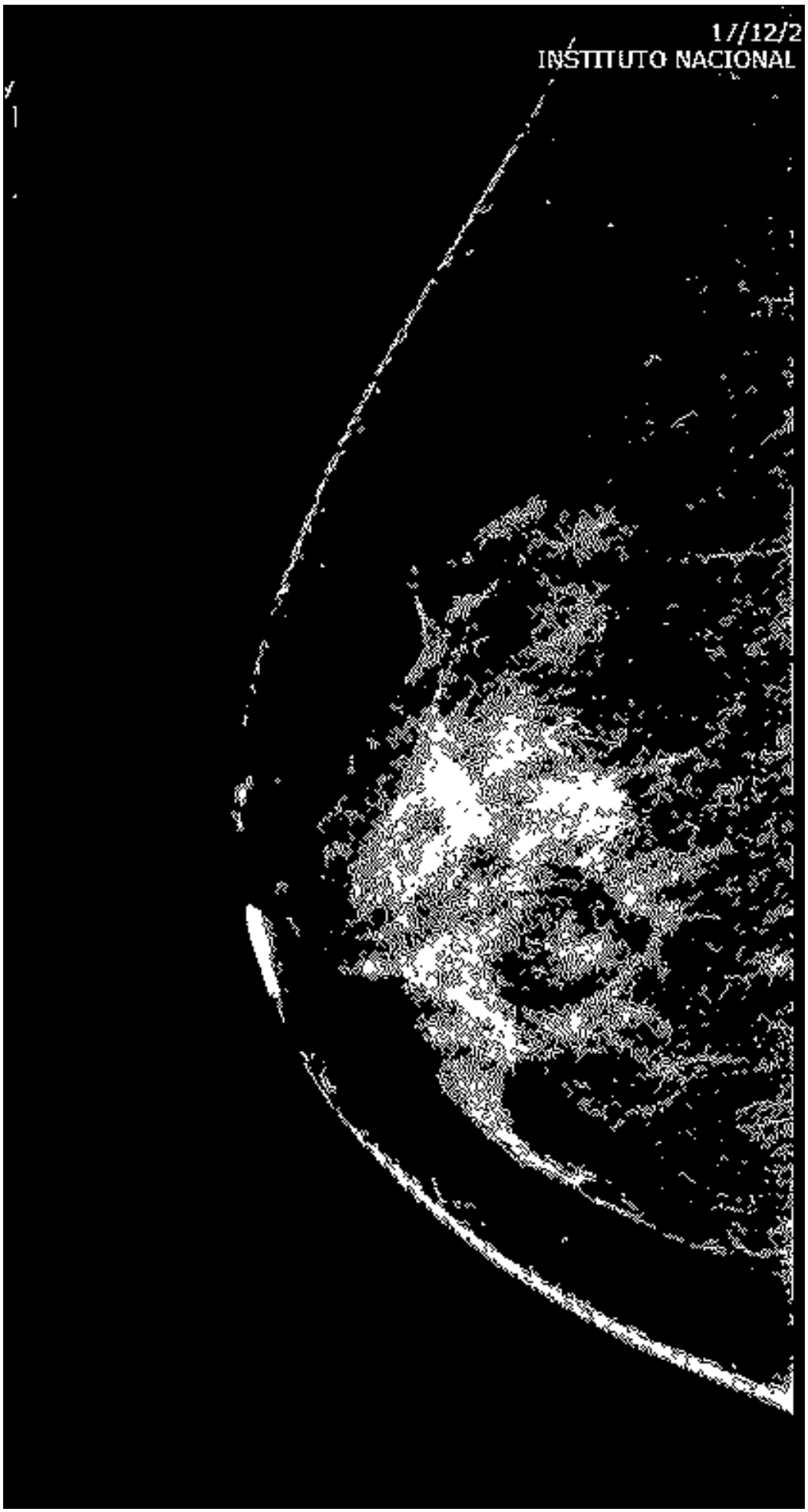
Para quitar el ruido final de la imagen, se utilizó el ejemplo de la práctica de la Dra Elisa en donde una de las imágenes se queda con ruidos y lineas pequeñas, que es más o menos lo que pasa con esta imágen.

In [24]:
```
import ssl
import requests
from PIL import Image, ImageDraw

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

n = nuevo
w, h = n.size
rgb = n.convert('RGB')
P = rgb.load()
negro = (0, 0, 0)
blanco = (255, 255, 255)
for f in range(h): # bordes verticales
    if P[0, f] == blanco:
        ImageDraw.floodfill(rgb, (0, f), negro)
    if P[w - 1, f] == blanco:
        ImageDraw.floodfill(rgb, (w - 1, f), negro)
for c in range(w): # bordes horizontales
    if P[c, 0] == blanco:
        ImageDraw.floodfill(rgb, (c, 0), negro)
    if P[c, h - 1] == blanco:
        ImageDraw.floodfill(rgb, (c, h - 1), negro)
n = rgb.convert('1')
P = n.load()
V = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
for f in range(1, h - 1): # sin bordes ahora
    for c in range(1, w - 1):
        if P[c, f] == 0:
            cont = 0
            for (df, dc) in V:
                if P[c + dc, f + df] == 0: # siempre existen
                    cont += 1
            if cont < 2:
                P[c, f] = 255 #  blanco
n.save('vecinos-m1.png')
vecinos = Image.open("vecinos-m1.png")
vecinos
```

Out[24]:

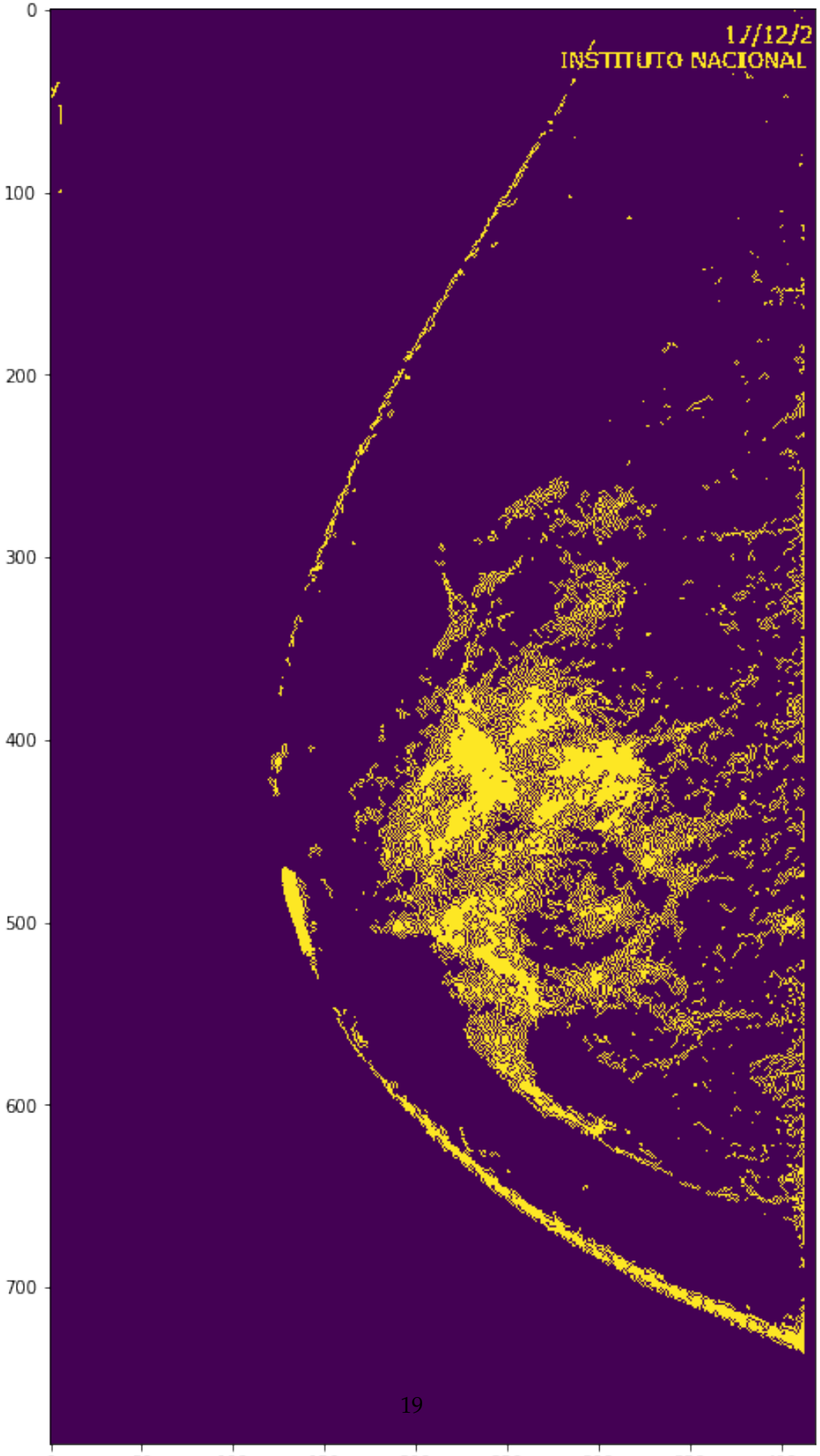

1//12/2
INSTITUTO NACIONAL

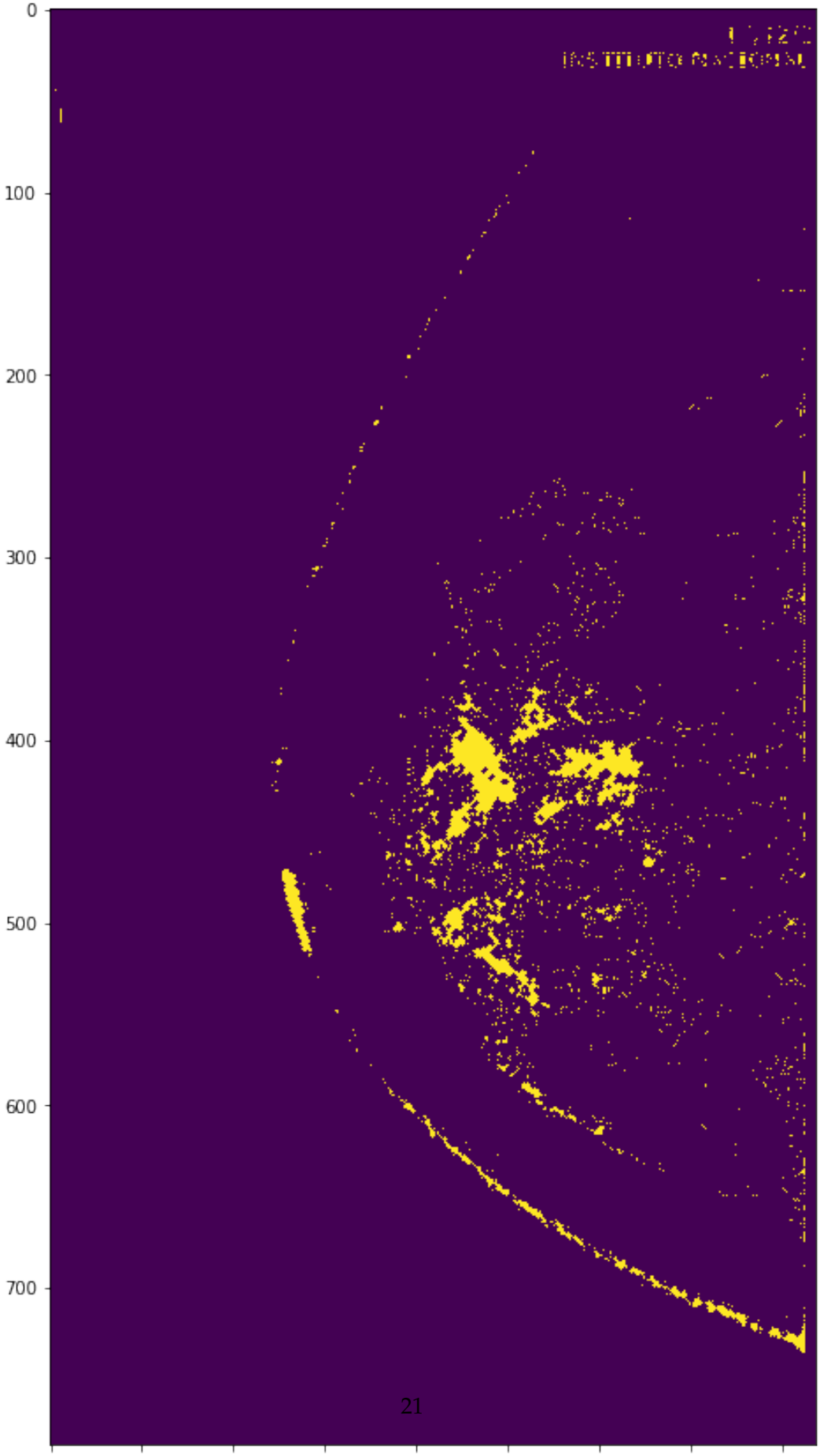
Después de hacer esto, es mucho más claro cuales son las zonas de más alta concentración. Ahora se cambia de herramienta de PIL a OpenCV, intentando sacar solo el contorno de las figuras en blanco que se ven en la imagen, logrando realizar una máscara.

```
In [34]: import cv2
         import numpy as np
         import matplotlib.pyplot as plt
         from skimage import io
         %matplotlib inline
         im = io.imread('vecinos-m1.png')
         plt.figure(figsize=(15,15))
         plt.imshow(im)


         # Contoured image
         ret,thresh = cv2.threshold(im, 120,255,cv2.THRESH_BINARY)
         contours = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)[-2]
         for contour in contours:
            cv2.drawContours(im, contour, -1, (0, 255, 0), 1)
         plt.figure(figsize=(15,15))
         plt.imshow(im)

Out[34]: <matplotlib.image.AxesImage at 0x1c2d560898>
```


17/12/2
INSTITUTO NACIONAL
0
100
200
300
400
500
600
700
0
50
100
150
200
250
300
350
400


0
100
200
300
400
500
600
700
0
50
100
150
200
250
300
350
400

Al final aún queda un poco de ruido en la parte derecha de la imagen, pero se puede apreciar mucho mejor la parte de la máscara que se quiere crear. Con esta última imagen se puede hacer una capa sobrepuesta de la imagen original para saber donde se encuentran las mayores concentraciones de pixeles, agilizando el trabajo de los expertos.