bootcamp 2023 slides

In [5]:

import cv2  
import numpy as np  
import pandas as pd  
from matplotlib import pyplot as plt  
import os

## SETUP[¶](#SETUP)

In [13]:

covid\_folder = os.path.join("..","data","COVID-19\_Radiography\_Dataset", "COVID")  
covid\_annotations\_folder = os.path.join("..","data","COVID-19\_Radiography\_Dataset", "COVID - Annotations")  
normal\_folder = os.path.join("..","data", "COVID-19\_Radiography\_Dataset","Normal")

## Reading Images[¶](#Reading-Images)

In [73]:

image\_file\_name = "COVID-1010.png"

In [21]:

co1010\_orig = cv2.imread(os.path.join(covid\_folder ,image\_file\_name ))  
co1010\_anno = cv2.imread(os.path.join(covid\_annotations\_folder ,image\_file\_name ))

In [29]:

fig, ax = plt.subplots(1,2, figsize=(20,10))  
ax[0].imshow(co1010\_orig)  
ax[1].imshow(co1010\_anno) ## [...,::-1]

Out[29]:

<matplotlib.image.AxesImage at 0x2fcd4eb0580>

![](data:image/png;base64;base64,)

## Difference[¶](#Difference)

In [33]:

diff = co1010\_orig - co1010\_anno  
# diff = diff[:,:,0]  
diff\_mean = diff.mean(axis=2)  
diff\_sum = diff.sum(axis=2)  
  
diff\_tr = cv2.threshold(diff,0,255,0) ## Treasholded  
diff\_tr = diff\_tr[1] ## take oly one layer!!

In [34]:

fig, ax = plt.subplots(1,2, figsize=(20,10))  
ax[0].imshow(diff[...,::-1]) ## Not Tresholded  
ax[1].imshow(diff\_tr) ## Tresholded

Out[34]:

<matplotlib.image.AxesImage at 0x2fcd4553d00>

![](data:image/png;base64;base64,)

## Manipulation[¶](#Manipulation)

In [56]:

kernel0 = cv2.getStructuringElement(cv2.MORPH\_RECT, (3,3))  
eroded = cv2.erode(diff\_tr.astype('uint8'), kernel0, iterations = 1)  
  
# dilated = cv2.dilate(eroded, kernel0)  
dilated = cv2.dilate(diff\_tr, kernel0, iterations=2)  
dilated5 = cv2.dilate(diff\_tr, kernel0, iterations=5)

In [58]:

fig, ax = plt.subplots(1,4, figsize=(28,10))  
ax[0].imshow(diff\_tr)  
ax[1].imshow(eroded)  
ax[2].imshow(dilated)  
ax[3].imshow(dilated5)

Out[58]:

<matplotlib.image.AxesImage at 0x2fcdba4a9a0>

![](data:image/png;base64;base64,)

## Floodfill[¶](#Floodfill)

#### (Floodfilling mask is NOT the final result)[¶](#Xe44bdf266a39441f6c6e9fd43d222c7c7649c9b)

In [64]:

height, width = co1010\_orig.shape[:-1]  
  
## A new mask for each floodfill opertion:  
mask\_ff0 = np.zeros((height + 2, width + 2), np.uint8)  
diff\_tr\_ff = cv2.floodFill(diff\_tr.astype('uint8').copy(), mask\_ff0, (1,1), (255,255,255)) ## Can I go without .copy() ?   
  
mask\_ff1 = np.zeros((height + 2, width + 2), np.uint8)  
eroded\_ff = cv2.floodFill(eroded.astype('uint8').copy(), mask\_ff1, (1,1), (255,255,255))  
  
mask\_ff2 = np.zeros((height + 2, width + 2), np.uint8)  
dilated\_ff = cv2.floodFill(dilated.astype('uint8').copy(), mask\_ff2, (1,1), (255,255,255))  
  
mask\_ff3 = np.zeros((height + 2, width + 2), np.uint8)  
dilated5\_ff = cv2.floodFill(dilated5.astype('uint8').copy(), mask\_ff3, (1,1), (255,255,255))

In [61]:

fig, ax = plt.subplots(1,4, figsize=(28,10))  
ax[0].imshow(diff\_tr\_ff[1])  
ax[1].imshow(eroded\_ff[1])  
ax[2].imshow(dilated\_ff[1])  
ax[3].imshow(dilated5\_ff[1])

Out[61]:

<matplotlib.image.AxesImage at 0x2fcdaebafa0>

![](data:image/png;base64;base64,)

### Invert the masks[¶](#Invert-the-masks)

In [65]:

diff\_tr\_ff = 255 - diff\_tr\_ff[1]  
eroded\_ff = 255 - eroded\_ff[1]  
dilated\_ff = 255 - dilated\_ff[1]  
dilated5\_ff = 255 - dilated5\_ff[1]  
fig, ax = plt.subplots(1,4, figsize=(28,10))  
ax[0].imshow(diff\_tr\_ff)  
ax[1].imshow(eroded\_ff)  
ax[2].imshow(dilated\_ff)  
ax[3].imshow(dilated5\_ff)

Out[65]:

<matplotlib.image.AxesImage at 0x2fcdcc1b160>

![](data:image/png;base64;base64,)

## Masking[¶](#Masking)

In [71]:

masked0 = cv2.bitwise\_and(co1010\_orig,co1010\_orig, mask=diff\_tr\_ff[:,:,1])  
masked1 = cv2.bitwise\_and(co1010\_orig,co1010\_orig, mask=eroded\_ff[:,:,1])  
masked2 = cv2.bitwise\_and(co1010\_orig,co1010\_orig, mask=dilated\_ff[:,:,1])  
masked3 = cv2.bitwise\_and(co1010\_orig,co1010\_orig, mask=dilated5\_ff[:,:,1])

In [72]:

fig, ax = plt.subplots(1,4, figsize=(28,10))  
ax[0].imshow(masked0)  
ax[1].imshow(masked1)  
ax[2].imshow(masked2)  
ax[3].imshow(masked3)

Out[72]:

<matplotlib.image.AxesImage at 0x2fcdfbcac70>

![](data:image/png;base64;base64,)

In [105]:

fig, ax = plt.subplots(1,4, figsize=(28,10))  
ax[0].imshow(co1010\_orig)  
ax[1].imshow(co1010\_orig)  
ax[2].imshow(masked0)  
ax[3].imshow(diff\_tr\_ff)  
  
for j in range(1,4):  
 for i in range(6):  
 ax[j].axhline(50 \* i, c="green")  
 ax[j].axvline(50 \* i, c="green")

![](data:image/png;base64;base64,)

## Dividing into Patches[¶](#Dividing-into-Patches)

In [84]:

## mask\_0 = getmask(image, annot) ## Homework!!  
  
  
image = co1010\_orig ## original image  
mask = diff\_tr\_ff[:,:,1] ## mask  
masked = masked0 ## maskedi image  
  
  
height, width = co1010\_orig.shape[:-1]  
path\_size = 16  
step\_size = 16 ## setting to 8 will produce overlapping patches  
  
patch\_directory = os.path.join("..","data","COVID-19\_Radiography\_Dataset","patches")  
  
i = 0  
while i < height - 128:  
 j = 0  
 while j < width - 128:  
 patch\_0 = image[i:path\_size+i, j:path\_size+j, :] ### ":" means "all channels"  
 patch\_m = mask [i:path\_size+i, j:path\_size+j ] ### " " means "0-th channel"  
  
 number\_of\_pixels\_from\_ROI = sum(sum((patch\_m//255)))  
 ratio\_of\_pixels\_from\_ROI = number\_of\_pixels\_from\_ROI // (path\_size\*path\_size)  
   
 if ratio\_of\_pixels\_from\_ROI > 0.25: ### if a patch overlaps with ROI by at least 25%  
 sample\_name = image\_file\_name.split(".")[0]  
 patch\_file\_name = sample\_name + "\_" + str(i) + "\_" + str(j) + ".tif"  
 patch\_file\_dir = os.path.join(patch\_directory, patch\_file\_name)  
 cv2.imwrite(patch\_file\_dir, patch\_0)   
  
 j += step\_size  
 i += step\_size

In [85]:

os.listdir(patch\_directory)

Out[85]:

['COVID-1010\_112\_112.tif',  
 'COVID-1010\_112\_80.tif',  
 'COVID-1010\_112\_96.tif',  
 'COVID-1010\_128\_64.tif',  
 'COVID-1010\_128\_80.tif',  
 'COVID-1010\_128\_96.tif',  
 'COVID-1010\_144\_64.tif',  
 'COVID-1010\_144\_80.tif',  
 'COVID-1010\_144\_96.tif',  
 'COVID-1010\_48\_112.tif',  
 'COVID-1010\_64\_112.tif',  
 'COVID-1010\_64\_96.tif',  
 'COVID-1010\_80\_112.tif',  
 'COVID-1010\_80\_80.tif',  
 'COVID-1010\_80\_96.tif',  
 'COVID-1010\_96\_112.tif',  
 'COVID-1010\_96\_80.tif',  
 'COVID-1010\_96\_96.tif',  
 'Thumbs.db']