Examen de aplazados

Alumno: Rolando Perez Olaguivel cod:20200197

Preguntas

1. Detección automática de grietas y fisuras en paredes (2pts)

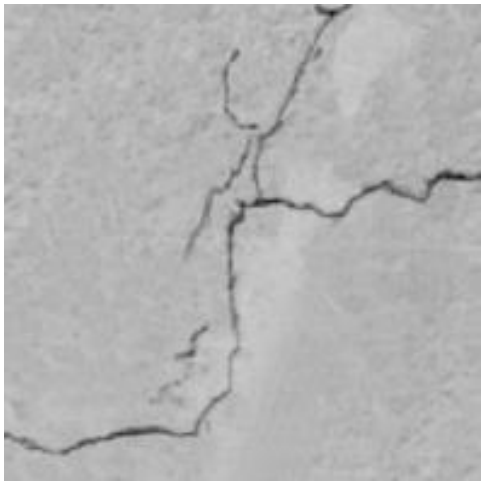


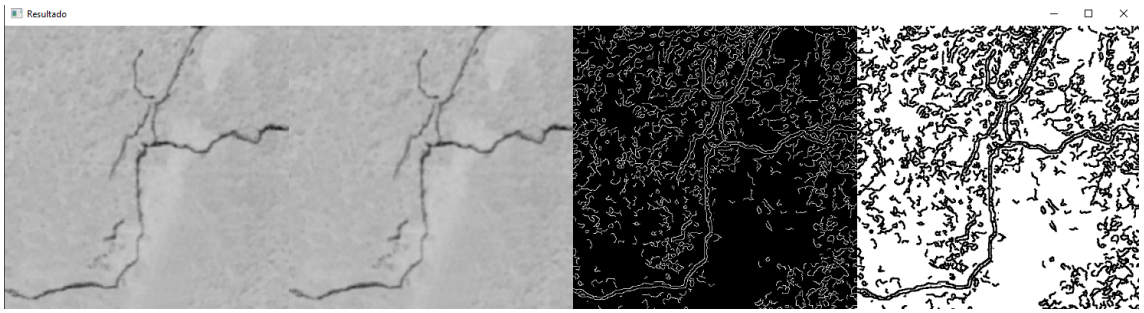
Ilustración 1 Salida

Solución:

```
import cv2
import numpy as np
from stackImages import StackImagen

# Carga la imagen
img = cv2.imread("imagenes1\pare_ejer1\pared5.jpg")

# Convertir la imagen a escala de grises
gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

## Convertir a HSL
#hsl_img = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)

# Aplicar un filtro Gaussiano para suavizar la
imagen
gris = cv2.GaussianBlur(gris, (3,3), 0)

# Aplicar filtro Canny para detectar bordes
borde = cv2.Canny(gris, 10, 25)

# Encontrar los contornos en la imagen
contours, hierarchy = cv2.findContours(borde,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

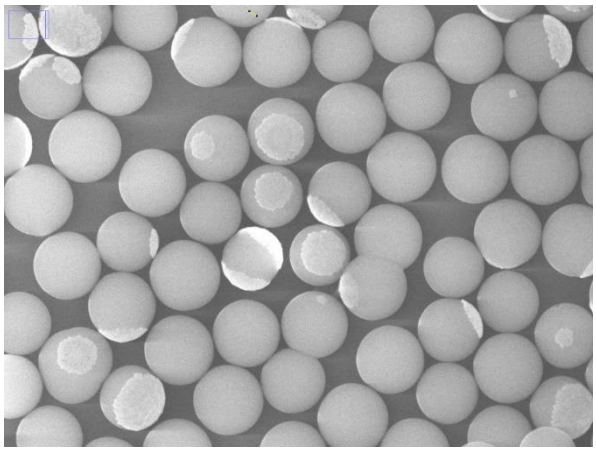
```
# Crear una imagen de fondo blanco
contorno = np.zeros((img.shape[0], img.shape[1], 3),
np.uint8)
contorno.fill(255)
# Dibujar los contornos en la imagen de fondo blanco
cv2.drawContours(contorno, contours, -1, (0, 0, 0),
2)
# Mostrar las imágenes resultantes creando un objeto
de la clase StackImagen
stack = StackImagen(0.7)

imagenes = [img,gris
            ,borde,contorno]

result = stack.stack_images(imagenes)
cv2.imshow("Resultado",result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Salida de consola:



Acá se puede ver como primero, se ve la imagen original, luego con escala de grises con el filtro gausiano (suavizar la imagen), luego la imagen de detección de bordes, y su inversa. Y con esa ya se tendría todos los recursos para detectar si hay una grieta o no en la pared.

2. Conteo automatico de colonias de bacterias (4pts)

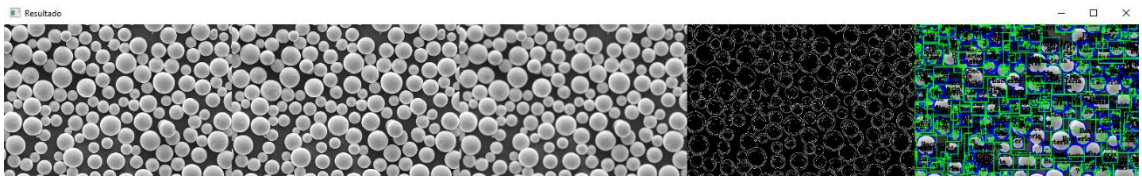


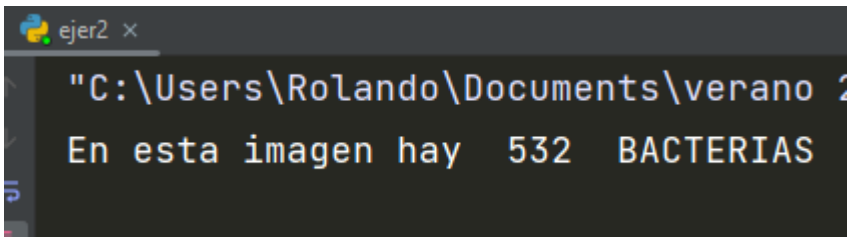
Solución:

```
import cv2
import numpy as np
from stackImages import StackImagen

#Cargar la imagen
img = cv2.imread(r"imagenes2\bacteria2.jpg")
img2 = img.copy()
# Convertir la imagen a escala de grises
gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Aplicar un filtro Gaussiano para suavizar la
imagen
gaus = cv2.GaussianBlur(gris, (7,7), 0)

# Aplicar filtro Canny para detectar bordes
borde= cv2.Canny(gaus,50,50)

#Funcion para obtener area de las figuras que salen
en la imagen
def getContorno(img):
    cont = 0
    contours, hierarchy = cv2.findContours(img,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    for cnt in contours:
        cv2.drawContours(img2,cnt,-1,(255,0,0),3)
        perimetro = cv2.arcLength(cnt,True)
        aprox =
```

```
cv2.approxPolyDP(cnt,0.02*perimetro,True)
        objcorner=len(aprox)
        x,y,w,h = cv2.boundingRect(aprox)
        if objcorner >4:
            b = "Bacteria"
            cont=cont+1
        else:
            b = "?"
        cv2.rectangle(img2,(x,y),(x+w,y+h),
(0,255,0),2)
        cv2.putText(img2, b,(x+(w//2)-18, y +
(h//2)-18), cv2.FONT_HERSHEY_COMPLEX,0.7,(0,0,0),2)
    print("En esta imagen hay ",cont," BACTERIAS")
#Llamamos a la funcion
getContorno(borde)

#Mostrar las imágenes resultantes creando un objeto
de la clase StackImagen
stack = StackImagen(0.4)
imagenes = [img,gris,gaus,img2,borde]


result = stack.stack_images(imagenes)
cv2.imshow("Resultado",result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

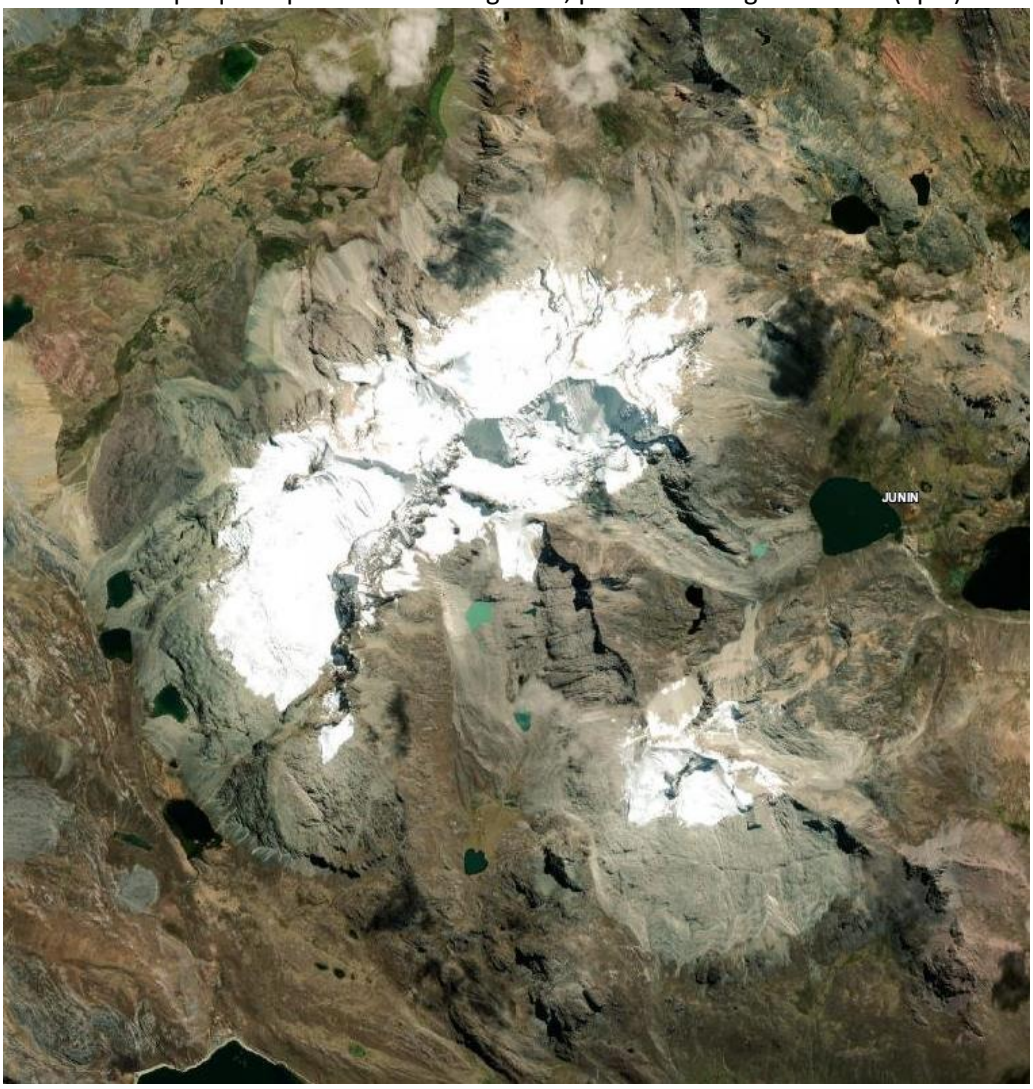
Salida de consola:





Como se puede observar, primero pone en escala de grises la imagen, luego la suavisa con el filtro gaussiano, se usa el filtro de Canny para detectar todos los bordes y al final cuenta cuantos de ellos son bacterias (en este caso circulos).

3. Extraer un shape que represente al área glaciar, puede usar segmentación (4pts)



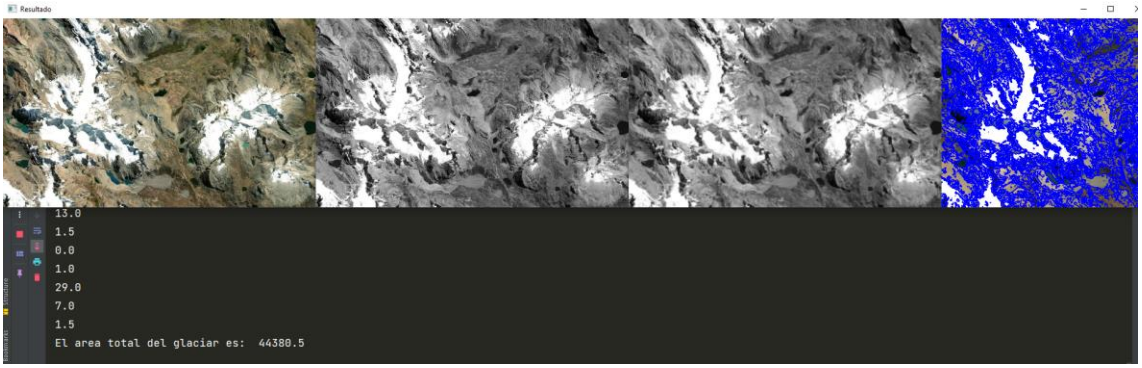
https://inaigem.maps.arcgis.com/apps/webappviewer/index.html?id=5379ac94516a4cb0a7f1cd0fa7bfc94b

Solución:

```
import cv2
import numpy as np

# Leer la imagen
from stackImages import StackImagen

img = cv2.imread(r'imagenes/glaciar.jpg')
img2 = img.copy()
# Convertir la imagen a escala de grises
gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Aplicar un filtro Gaussiano para suavizar la
imagen
```

```
gaus = cv2.GaussianBlur(gris, (7,7), 0)

# Aplicar filtro Canny para detectar bordes
borde= cv2.Canny(gaus,50,50)

#Funcion para obtener area de las figuras que salen
en la imagen
def getContorno(img):
    cont = 0
    contours, hierarchy = cv2.findContours(img,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    for cnt in contours:
        area = cv2.contourArea(cnt)
        print(area)
        cv2.drawContours(img2,cnt,-1,(255,0,0),3)
        cont = cont +area
    print("El area total del glaciar es: ",cont)
#Llamamos a la funcion
getContorno(borde)

#Mostrar las imágenes resultantes creando un objeto
de la clase StackImagen
stack = StackImagen(0.4)
imagenes = [img,gris,gaus,img2]


result = stack.stack_images(imagenes)
cv2.imshow("Resultado",result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

salida de consola:



En este caso en las imágenes podemos observar primero la imagen original, luego en escala de grises, luego suavizado, y al final con todos los bordes resaltados, y al final nos muestra el área que encierran dichos bordes.

4. Detección de vehículos en un video (7pts)

2011_09_26_drive_0001 (0.4 GB)
Length: 114 frames (00:11 minutes)
Image resolution: 1392 x 512 pixels
Labels: 12 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 2 Cyclists, 1 Trams, 0 Misc
Downloads: [unsynced+unrectified data] [synced+rectified data] [calibration] [tracklets]

http://www.cvlibs.net/datasets/kitti/raw_data.php

Solución:

```
import os
import re
import cv2 # opencv library
import numpy as np
from os.path import isfile, join
import matplotlib.pyplot as plt


#VIdeo importado
vid = cv2.VideoCapture(r'recursos\video.mp4')

#tamaño minimo del objeto
min_width_react=80
min_hight_react=80

count_line_postion = 550 #Para asignar la posicion
de la linea


#Inicializando el Subestructurado, trata de quitar
el fondo y enfocar el objeto en movimiento
algo = cv2.bgsegm.createBackgroundSubtractorMOG()

def center_handle(x,y,w,h):
    x1=int(w/2)
    y1=int(h/2)
    cx = x+x1
    cy = y + y1
    return cx,cy

detect = []
offset = 6 #error permitido entre píxeles
cont = 0

while True:
    #La función ret, frame1 = vid.read() devuelve
```

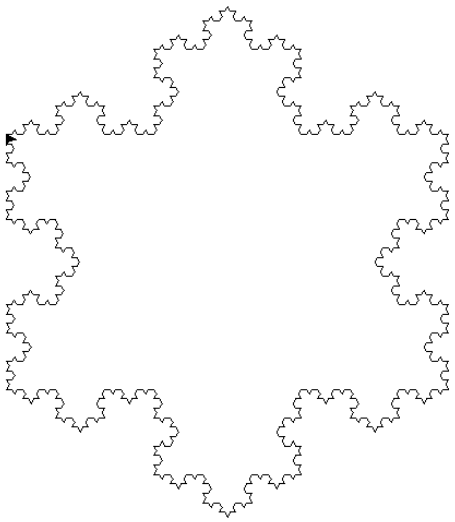
```
dos valores:
    # ret es un valor booleano que indica si el
fotograma se ha leído correctamente,
    # y frame1 es una matriz NumPy que representa el
fotograma.
    ret, frame1 = vid.read()

    #Ponindo el frame en escala de grises
    gris = cv2.cvtColor(frame1,cv2.COLOR_BGR2GRAY)

    #Suavisando la imagen con el filtro Gausianico
    gaus = cv2.GaussianBlur(gris,(3,3),5)

    #Aplicamos a cada frame el Substructurado
    img_sub = algo.apply(gaus)

    #
    dilat = cv2.dilate(img_sub,np.ones((5,5)))

    #
    kernel =
cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))

    #
    dilatada =
cv2.morphologyEx(dilat,cv2.MORPH_CLOSE, kernel)
    dilatada = cv2.morphologyEx(dilatada,
cv2.MORPH_CLOSE, kernel)

    #
    counterShape,h = cv2.findContours(dilatada,
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    cv2.line(frame1,(25,count_line_postion),(1200,
count_line_postion),(255,127,0),3)


    for (i,c) in enumerate(counterShape):
        (x,y,w,h) = cv2.boundingRect(c)
        validate_counter = (w>= min_width_react) and
(h>= min_hight_react)
        if not validate_counter:
            continue
```

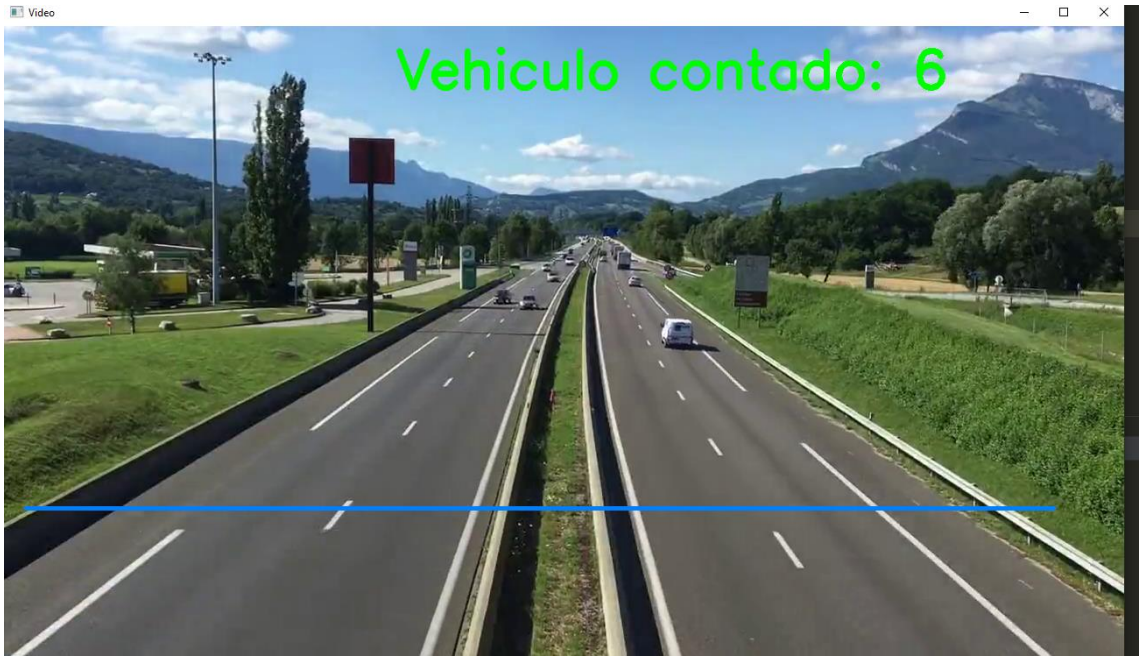
```
cv2.rectangle(frame1,(x,y),(x+w,y+h),(0,0,255),2)
        center = center_handle(x,y,w,h)
        detect.append(center)
        cv2.circle(frame1, center, 4, (0,255,0), -1)

        for(x,y) in detect:
            if y < (count_line_postion+offset) and y
> (count_line_postion-offset):
                cont= cont+1

cv2.line(frame1,(25,count_line_postion),(1200,
count_line_postion),(0,127,255),3)
            detect.remove((x,y))
            print("vehiculo contado: ", cont)

    cv2.putText(frame1,"Vehiculo contado:
"+str(cont),(450,70),cv2.FONT_HERSHEY_SIMPLEX,2,(0,2
55,0),5)




    cv2.imshow('Video',frame1) #muestar el fotograma

    if cv2.waitKey(1) == 13:
        break

cv2.destroyAllwindows()
vid.release()
```

5. Modele en 3d un carro o un objeto del hogar (use polígonos), póngale una textura (3pts)
6. Dibuje un fractal con codigo (2pts)
   Solución:

```
from turtle import*

def f(n):
    speed(0.1)
    if n==0:
        forward(5)
    else:
        f(n-1)
        left(60)
        f(n-1)
        right(120)
        f(n-1)
        left(60)
        f(n-1)
def h(n):
    f(n)
    right(120)
    f(n)
    right(120)
    f(n)

h(4)
turtle.done()
```

Salida de consola:





Como se puede observer, este código nos muestra el video con una línea, esta línea sirve como contador de los autos que van pasando sobre ella.

Como se puede observar este código recursivo, sirve para hacer el fractal de copo de nieve.