Asignatura	Datos Equipo 34	Fecha
Percepción Computacional	Torres Plácido Felipe	**19/07/2021**
	Victor Eduardo Anchondo Trejo	

Actividad uso de filtros espaciales y morfológicos

Objetivo:

El objetivo de este trabajo es buscar y analizar los filtros espaciales y morfológicos más habitualmente usados en la literatura. Esto permitirá consolidar los conceptos y operaciones sobre imágenes aprendidas en el tema 7.

Uso práctico: Ver-de todo

Ver-de todo es la iniciativa para ayudar a mejorar la calidad del aire en la ciudad de México y zona metropolitana con ayuda de Inteligencia Artificial seremos capaces de enfocar los esfuerzos en reforestación a pequeña escala que tendrá un impacto directo en la calidad de vida de los habitantes de la ciudad.

Ver-de todo siempre respetará la privacidad de los habitantes y sus propiedades por ese motivo el índice será por cuadra y nunca hará referencia a sus propietarios o cualquier dato sensible.

Ver-de todo contendrá una tienda de especies que invitará a los usuarios y proveedores a conservar la fauna nativa de la región.

Verde-todo tendrá opción de donar una planta para lugares que más lo necesitan y empleara a personas que lo requieran respetando todas prestaciones de seguridad social por que el centro de ver-de todo son las personas y su bienestar.

Asignatura	Datos Equipo 34	Fecha
Percepción Computacional	Torres Plácido Felipe	**19/07/2021**
	Victor Eduardo Anchondo Trejo	

Problema: Contabilizar arboles mediante uso de filtros morfológicos.

Para conseguir realizar la actividad tendremos en cuenta los principales filtros morfológicos y sus características buscando dar una solución comprobable de la aplicación de los conocimientos.

Para esto tuvimos que aplicar las siguientes operaciones:

1. Filtrado de elementos de color verde.
2. Operación de Apertura
    - Operador Morfológico de Erosión.
    - Operador Morfológico de Dilatación.
3. Contar los elementos en la imagen procesada.

Ventajas.

Al hacer uso de imágenes de Google Maps se cuenta con la posibilidad de contar con las imágenes de prácticamente todas las zonas urbanas por lo que obtendremos las imágenes de **Google Earth Pro** y presentamos la iniciativa a para realizar pruebas en la plataforma Google Earth Engine (pendiente de aprobación).

Desventajas.

El proyecto por sí mismo requiere una inversión por lo que los recursos financieros pueden llegar a ser un tema así mismo la complejidad de la iniciativa puede que sea el principal reto ante un equipo de personas tan reducido.

Resultados obtenidos.

Para realizar el primer paso: **Filtrado de elementos de color verde**, tuvimos que convertir la imagen leída a una modelo de colores HSV (ECURED, 2021), luego utilizando el rango inferior [40,40,40] y superior [70,255,255] para el color verde pudimos obtener una mascara de la imagen con todos los elementos que están en ese rango de color verde. Referencia obtenida de los documentos de OpenCV (OpenCV, 2021).

Para el segundo paso: **Operación de Apertura**, se utilizó la combinación del **operador de Erosión** seguido del **operador de Dilatación**, esto con el fin de intentar separar los objetos que puedan estar solapados entre si. Para ambos operadores se hizo uso de las funciones de ***erode()*** y ***dilate()*** de la librería de OpenCV (OpenCV, 2021).

Para el tercer paso: **Contar los elementos en la imagen procesada**, se utilizó la función ***findContours()*** de la Liberia de OpenCV (OpenCV, 2021) y a esto le agregamos funcionalidad extra ya que hay casos donde los arboles están muy cerca uno de otro y la función regresa resultados erróneos, para esto utilizamos la siguiente función:

```
def countObjects(img):
    counters = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    counters = counters[0] if len(counters) == 2 else counters[1]
    treeCount = 0
    for c in counters:
        area = cv2.contourArea(c)
        if area > 50:
            x,y,w,h = cv2.boundingRect(c)
            cv2.drawContours(img, [c], -1, (36,255,12), 2)
            treeCount += 1

    return treeCount
```

(Lógica obtenida de: https://stackoverflow.com/questions/58509026/counting-special-elements-on-image-with-opencv-and-python)

Lógica principal.

Inicia lógica principal

M↓

```
# Obtiene e imprime la lista de imagenes jpeg dentro del directorio pictures/
picturePathList = getTestPicturesInPath("pictures/", ".jpeg")
print("Imagenes a procesar: " + str(len(picturePathList)))
```

```
Imagenes a procesar: 6
```

M↓

```
# Ciclo para procesar cada una de las imagenes
for picturePath in picturePathList:
    # Se lee la imagen del mapa con sus colores originales
    originalImage = cv2.imread(picturePath, cv2.IMREAD_UNCHANGED)

    # Extraer el color verde
    greenMask = getGreenMask(originalImage)

    # Tamaño del elemento estructural para aplicar erosion y dilatacion
    structuralElementSize = 5

    # Erosion
    imgEroded = erodeImage(greenMask, structuralElementSize)

    # Dilatacion
    imgDilated = dilateImage(imgEroded, structuralElementSize)

    # Contar
    count = countObjects(imgDilated)

    # Agrega un titulo al plot
    addTitleToPlot()

    # Se prepara cada imagen para ser mostrada
    addImageToPlot(originalImage, "Original", 231)
    addImageToPlot(greenMask, "Mascara de Verde", 232)
    addImageToPlot(imgEroded, "Erosion", 233)
    addImageToPlot(imgDilated, "Dilatacion", 234)

    # Se muestra una ventana con todas las imagenes de los pasos aplicados
    plt.show()
```

Asignatura	Datos Equipo 34	Fecha
Percepción Computacional	Torres Plácido Felipe	**19/07/2021**
	Victor Eduardo Anchondo Trejo	

Funciones principales.

Función que regresa la máscara del color verde en HSV de una imagen

```
def getGreenMask(originalImage):
    # Se convierte la imagen original a colores HSV
    hsv = cv2.cvtColor(originalImage, cv2.COLOR_BGR2HSV)

    # Se define el rango del color verde en HSV
    lowerGreenLimit = np.array([40,40,40])
    upperGreenLimit = np.array([70,255,255])

    # Se aplica el rango a la imagen hsv y se obtiene la imagen del color verde
    greenMask = cv2.inRange(hsv, lowerGreenLimit, upperGreenLimit)

    return greenMask
```

Función que regresa una imagen a la que se le ha aplicado el operador morfológico de erosión

```
def erodeImage(image, structuralElementSize):
    # Creación del kernel
    kernel = np.ones((structuralElementSize, structuralElementSize), np.uint8)
    # Erosionar imagen usando cv2.erode()
    img = cv2.erode(image, kernel, iterations=1)

    return img
```

Función que regresa una imagen a la que se le ha aplicado el operador morfológico de erosión

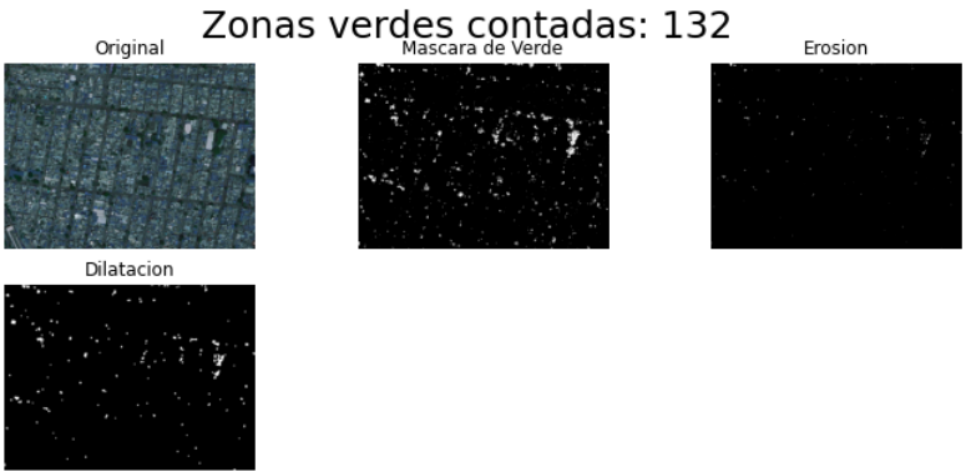
```
def erodeImage(image, structuralElementSize):
    # Creación del kernel
    kernel = np.ones((structuralElementSize, structuralElementSize), np.uint8)
    # Erosionar imagen usando cv2.erode()
    img = cv2.erode(image, kernel, iterations=1)

    return img
```

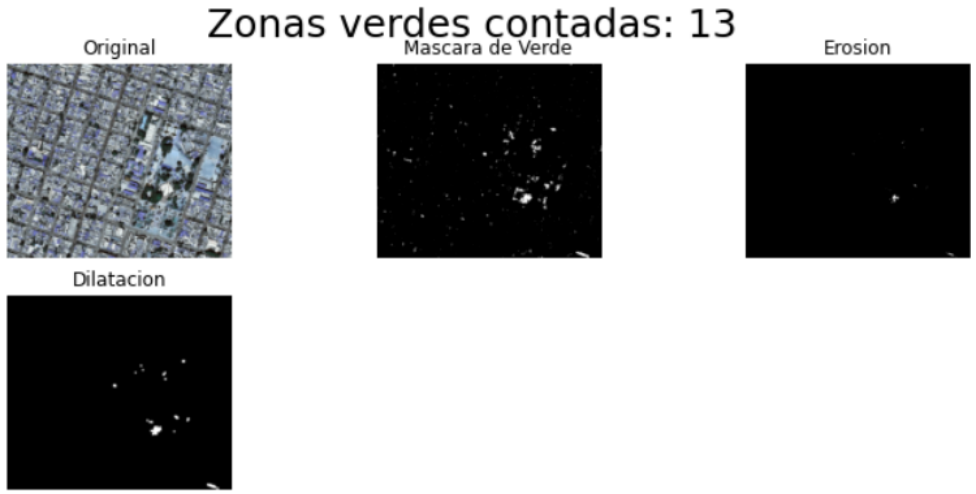
Función que regresa una imagen a la que se le ha aplicado el operador morfológico de erosión

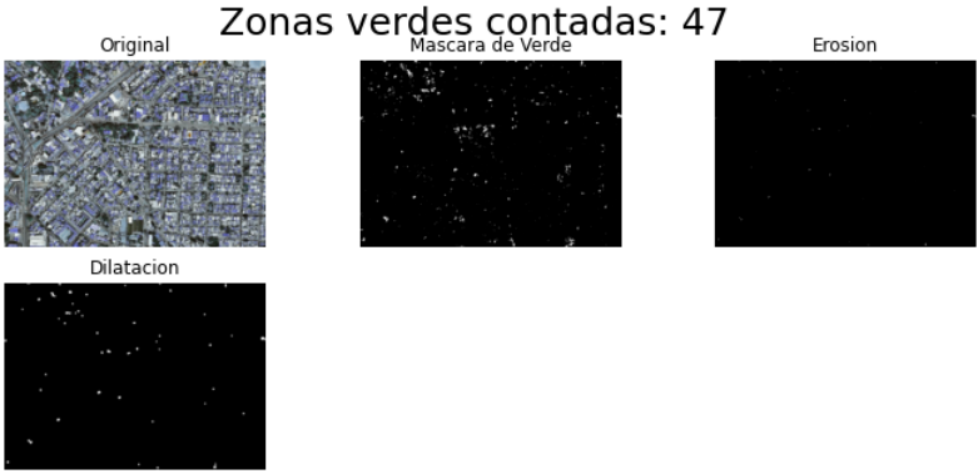
```
def erodeImage(image, structuralElementSize):
    # Creación del kernel
    kernel = np.ones((structuralElementSize, structuralElementSize), np.uint8)
    # Erosionar imagen usando cv2.erode()
    img = cv2.erode(image, kernel, iterations=1)

    return img
```

Asignatura	Datos Equipo 34	Fecha
Percepción Computacional	Torres Plácido Felipe	**19/07/2021**
	Victor Eduardo Anchondo Trejo	

Resultados obtenidos.







Referencias.

ECURED. (19 de July de 2021). *Modelo HSV*. Obtenido de ECURED: https://www.ecured.cu/Modelo_HSV

OpenCV. (19 de July de 2021). *Changing Color-space*. Obtenido de OpenCV: https://docs.opencv.org/4.5.2/df/d9d/tutorial_py_colorspaces.html

OpenCV. (19 de July de 2021). *OpenCV: Eroding and Dilating*. Obtenido de OpenCV: https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

OpenCV. (19 de July de 2021). *Structural Analysis and Shape Descriptors*. Obtenido de OpenCV: https://docs.opencv.org/4.5.2/d3/dc0/group__imgproc__shape.html#gadf1ad6a0b82947fa1fe3c3d497f260e0