Proyecto Final de Reconocimiento de Patrones. Detección de Sargazo

Aguilar Luna Gabriel Daniel, Rodríguez Agiss Zuriel Uzai

***Abstract*—En este proyecto se busca apoyar a la solución al problema de sargaso en la costa del caribe en México por medio de un sistema que efectúa parte de los temas vistos en la materia de reconocimiento de patrones para la detección del sargaso en imágenes. Esto se llevará a cabo por medio de la caracterización y clasificación de texturas**

***Index Terms*—texturas, k-means, matriz de coocurrencias, hsv, entropía, energía, sargazo**

I. Objetivo

Se espera que se realice análisis de las imágenes para obtener la tareaa de clasificar la escena tomando como prioridad el sargazo

II. Introduction

El sargazo ha representado un problema para las poblaciones de Cancún y Yucatán principalmente, e incluso ha incrementado en los últimos meses en la región del Caribe, y se apunta a niveles incluso similares a los de 2018, cuando supuso una pesadilla para la actividad turística.

Un reporte de julio de 2020 elaborado por las secretarías de Turismo y de Medio Ambiente y de Recursos Naturales, entre otras dependencias, advirtió que el sargazo ocasionó que comunidades de pastos marinos fueran reemplazadas por algas calcáreas entre 2014 y 2015, lo que tuvo como resultado una pérdida de la biomasa debajo de la superficie cubierta por las algas estimada entre 61.6 y 99.5

Fig. 1. Sargazo en playas de Cancún, Quintana Roo

La amenaza del sargazo ha llegado a tal grado que se nos encomendó ayudar a la Red de Monitoreo del Sargazo con un sistema de detección de sargazo mediante el reconocimiento de patrones, la cuál es una herramienta para informar y alertar sobre del arribo masivo del sargazo en playas de Cancún y Riviera Maya. Para ello se hará uso de los siguientes tópicos:

A. Análisis de texturas

El analisis de texturas hace referencia a la caracterización de las regiones de una imagen por su contenido de textura. El analisis de texturas intenta cuantificar las cualidades intuitivas descritas por terminos como ´ aspero, suave, sedoso o accidentado en funcion de la variaci ´ on espacial en las intensidades de píxeles. En este sentido, la rugosidad o bache se refiere a variaciones en los valores de intensidad, o niveles de gris. El analisis de texturas se utiliza en varias aplicaciones, incluyendo la teledeteccion, la inspecci ´ on automatizada y el procesamiento de imagenes médicas. El aná alisis de texturas se puede utilizar para encontrar los l´ımites de textura, denominados segmentacion de texturas. El análisis de texturas puede ser util cuando los objetos de una imagen se caracterizan más por su textura que por la intensidad, y las tecnicas de umbral tradicionales no se pueden utilizar de forma eficaz. Es posible usar metodos estadísticos para el analisis de texturas como por ejemplo:

- Analizar la distribucion espacial de valores de gris es una calidad que define la textura.
- Analizar la distribucion espacial de los valores de gris, se computan caracter´ısticas locales de la textura. Ejemplos:
  - Media y varianza.
  - Co-ocurrencia y diferencias de niveles de gris.

B. Clasificador K-Means

El algoritmo Kmeans es un algoritmo iterativo que intenta dividir el conjunto de datos en subgrupos (clústeres) distintos no superpuestos definidos previamente por K donde cada punto de datos pertenece a un solo grupo. Intenta hacer que los puntos de datos intra-clúster sean lo más similares posible y al mismo tiempo mantiene los clústeres lo más diferentes (lejos) posible. Asigna puntos de datos a un grupo de modo que la suma de la distancia al cuadrado entre los puntos de datos y el centroide del grupo (media aritmética de todos los puntos de datos que pertenecen a ese grupo) es mínima. Mientras menos variación tengamos dentro de los conglomerados, más homogéneos (similares) serán los puntos de datos dentro del mismo conglomerado.

C. Superpixeles

Las técnicas de superpíxeles segmentan una imagen en regiones considerando medidas de similitud definidas mediante características perceptivas. Es decir, a diferencia de las cuencas hidrográficas y MSER, las técnicas de superpíxeles crean grupos de píxeles que se ven similares. La motivación es obtener regiones que representen descripciones significativas con muchos menos datos que cuando se utilizan todos los píxeles de una imagen. La premisa es que al reducir el número de primitivas, se reduce la redundancia reduciendo así la complejidad de las tareas de reconocimiento. Además, los superpíxeles reemplazan la estructura rígida de píxeles al delinear regiones que mantienen el significado en la imagen, por lo que las regiones brindan información sobre la estructura de la escena, lo que hace que otras tareas de procesamiento sean más simples que usar píxeles de una sola imagen. En general, las técnicas de superpíxeles se basan en medidas que buscan similitudes de color y medidas de la forma de las regiones. El proceso de segmentación también incorpora bordes o fuertes cambios de intensidad para delinear regiones.

III. Desarrollo

Para el desarrollo de este proyecto se cuenta con el siguiente tipo de imágenes a procesar

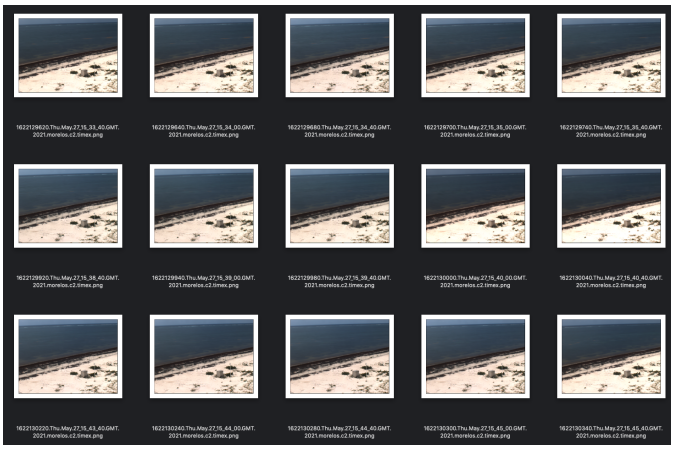

Fig. 2. Muestra de ejemplo de imágenes a ser usadas para entrenamiento y prueba

Lo primero que se tuvo que hacer fue un preprocesamiento en la que se decidió usar el espectro HSV de la imagen en lugar de la imagen original por la información valiosa que ofrece

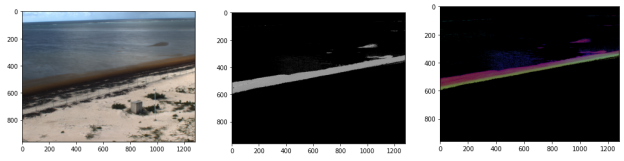

Fig. 3. Al inicio se contempló transformar la imagen al espectro de gris, pero se observó que se perdía información relevante por lo que usamos los valores HSV

Posteriormente se procedió a calcular los superpixeles, pero en nuestras imágenes de espectro HSV.

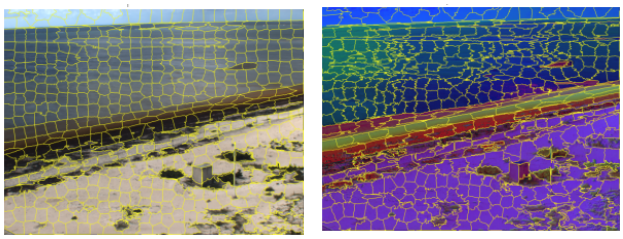

Fig. 4. Superpíxeles generados automáticamente. Se generaron también en la imagen original con fines meramente de contraste

Se generan las matrices de co-ocurrencia y se calcula la entropia y energía

$$Energía = \sum_{i,j=1}^{N} (P_{ij})^2$$

$$Entropía = \sum_{i,j=1}^{N} P_{ij}(i-j)^2$$

Y para finalizar, se realiza la clasificación con k-means

IV. Resultado

A continuación se aprecian nuestros resultados obtenidos. En las imágenes se debe apreciar que el color azul representa a nuestro sargazo, y que si bien no diferencía entre la playa y la arena, es lo deseado ya que solo nos interesa la clase de sargazo.

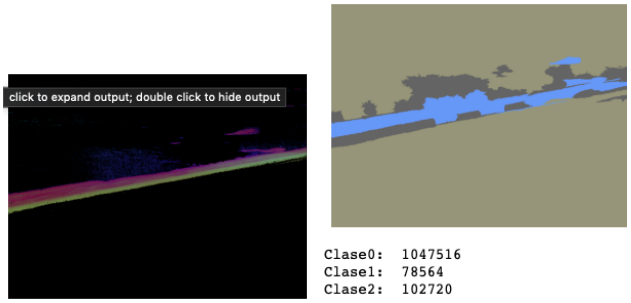

Fig. 5. Probando con la imagen de la izquierda se obtiene el resultado de la derecha incluyendo la cantidad de píxeles

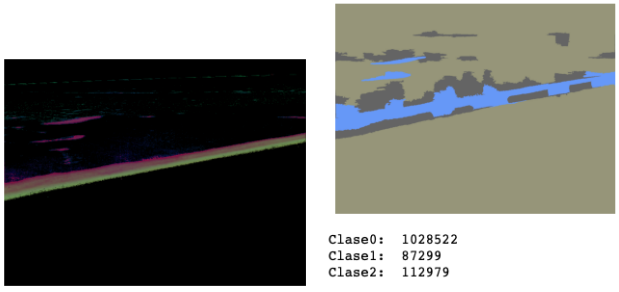

Fig. 6. Probando con la imagen de la izquierda se obtiene el resultado de la derecha incluyendo la cantidad de píxeles

También se hizo un análisis de movimiento del sargazo basándonos en una secuencia de dos imágenes, y obtuvimos el siguiente resultado

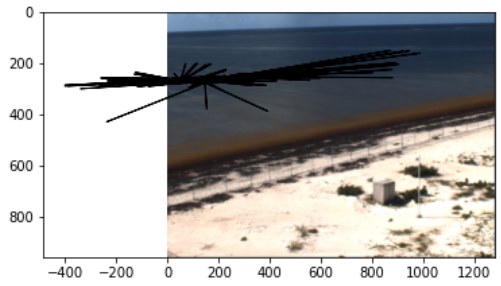

Fig. 7. Se aprecia que hay un movimiento horizontal de izquierda a derecha

V. Código

```
import matplotlib.pyplot as plt
import skimage as ski
import numpy as np
import imageio as io
from shapely.geometry import Polygon, MultiPolygon
from descartes import PolygonPatch
from math import log
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from skimage.segmentation import slic
from skimage.segmentation import mark_boundaries
import matplotlib.image as pltI
import os
import cv2

# COM horizaontal
def sp_COMMaker_h(imagen, marker, label_n, norm = False):
    matriz_auxiliar = np.zeros((np.amax(imagen)+1,)*2, dtype=int)
    dimensiones = imagen.shape
    normalizador = 0
    for i in range(dimensiones[0]):
        for j in range(dimensiones[1]-1):
            if (marker[i][j] == label_n and marker[i][j+1] == label_n):
                matriz_auxiliar[imagen[i][j]][imagen[i][j+1]] += 1
                normalizador += 2
    COM = matriz_auxiliar + np.transpose(matriz_auxiliar)
    return (COM/normalizador) if norm else COM

# COM vertical
def sp_COMMaker_v(imagen, marker, label_n, norm = False):
    matriz_auxiliar = np.zeros((np.amax(imagen)+1,)*2, dtype=int)
    dimensiones = imagen.shape
    normalizador = 0
    for i in range(dimensiones[0]-1):
        for j in range(dimensiones[1]):
            if (marker[i][j] == label_n and marker[i+1][j] == label_n):
                matriz_auxiliar[imagen[i][j]][imagen[i+1][j]] += 1
                normalizador += 2
    COM = matriz_auxiliar + np.transpose(matriz_auxiliar)
    return (COM/normalizador) if norm else COM

# COM diagonal (esq.inf.izq -> esq.sup.der.)
def sp_COMMaker_45(imagen, marker, label_n, norm = False):
    matriz_auxiliar = np.zeros((np.amax(imagen)+1,)*2, dtype=int)
    dimensiones = imagen.shape
    normalizador = 0
    for i in range(1,dimensiones[0]):
        for j in range(dimensiones[1]-1):
            if (marker[i][j] == label_n and marker[i-1][j+1] == label_n):
                matriz_auxiliar[imagen[i][j]][imagen[i-1][j+1]] += 1
                normalizador += 2
    COM = matriz_auxiliar + np.transpose(matriz_auxiliar)
    return (COM/normalizador) if norm else COM

# COM diagonal (esq.sup.izq. -> esq.inf.der.)
def sp_COMMaker_135(imagen, marker, label_n, norm = False):
    matriz_auxiliar = np.zeros((np.amax(imagen)+1,)*2, dtype=int)
    dimensiones = imagen.shape
```

```
    normalizador = 0
    for i in range(1,dimensiones[0]):
        for j in range(1,dimensiones[1]):
            if (marker[i][j] == label_n and marker[i-1][j-1] == label_n):
                matriz_auxiliar[imagen[i][j]][imagen[i-1][j-1]] += 1
                normalizador += 2
    COM = matriz_auxiliar + np.transpose(matriz_auxiliar)
    return (COM/normalizador) if norm else COM

def entropy(COM_n):
    entropia = 0
    for renglon in COM_n:
        for value in renglon:
            entropia += (value*log(value)) if (value>0) else 0
    return -entropia

def homogeneity(COM_n):
    homogeneidad = 0
    Y, X = COM_n.shape
    for i in range(Y):
        for j in range(X):
            homogeneidad += COM_n[i][j]/(1+abs(i-j))
    return homogeneidad

#Angular Second Moment, Energy
def smoothness(COM_n):
    ASM = 0
    Y, X = COM_n.shape
    for i in range(Y):
        for j in range(X):
            ASM += COM_n[i][j]**2
    return ASM

def contrast(COM_n,k=2,n=1):
    contraste = 0
    Y, X = COM_n.shape
    for i in range(Y):
        for j in range(X):
            contraste += ((i-j)**k)*(COM_n[i][j]**n)
    return contraste

def sp_COMs(imagen, marker):
    vcs_COM = []
    for nSP in np.unique(marker):
        COM_h = sp_COMMaker_h(rgb2gray(imagen), marker, nSP, True)
        #COM_v = sp_COMMaker_v(rgb2gray(imagen), marker, nSP, True)
        vcs_COM.append([entropy(COM_h), smoothness(COM_h)])
    return vcs_COM

def rgb2gray(imagen):
    gris = ((imagen[:,:,0]+imagen[:,:,1]+imagen[:,:,2])/3).astype(int)
    return gris

def mostrarSP(superpixeles, clasi, colores, imagen, guardar='aux'):
    superpixel_aux = np.zeros(imagen.shape, dtype=np.uint8)
    Y,X = superpixeles.shape
    for i in range(Y):
        for j in range(X):
            superpixel_aux[i,j] = colores[clasi[superpixeles[i,j]-1]]
    plt.imshow(superpixel_aux)
    plt.axis('off')
    plt.show()
    pltI.imsave('./resultados/Kmeans'+guardar+'.png', superpixel_aux)

def get_SP(imagen, nombre='', **kwargs):
    show = (kwargs["show"] if ("show" in kwargs) else False)
    save = (kwargs["save"] if ("save" in kwargs) else False)
    i_sp = slic(imagen, n_segments=600, start_label=1, compactness=8.0, sigma=1.0)
    if show:
        plt.imshow(mark_boundaries(imagen, i_sp))
        plt.axis('off')
        plt.title(nombre, color='darkgray')
        plt.show()
    if save:
        pltI.imsave('sp/sp'+nombre+'_'+str(np.amax(i_sp))+'_n600_c8_s1'+nombre[11:21]+'.png', mark_boundaries(imagen, i_sp))
    return i_sp

def get_SP_hsv(imagen, nombre='', **kwargs):
    show = (kwargs["show"] if ("show" in kwargs) else False)
    save = (kwargs["save"] if ("save" in kwargs) else False)
    i_sp = slic(imagen, n_segments=600, start_label=1, compactness=8.0, sigma=1.0)
    if show:
        plt.imshow(mark_boundaries(imagen, i_sp))
        plt.axis('off')
        plt.title(nombre, color='darkgray')
        plt.show()
    if save:
        pltI.imsave('sp/sp_'+nombre+'HSV_'+str(np.amax(i_sp))+'_n600_c8_s1'+nombre[11:21]+'.png', mark_boundaries(imagen, i_sp))
    return i_sp
```

VI. Conclusiones

VII. Referencias

- "El sargazo ensombrece la recuperación del turismo en el Caribe mexicano" (29 de junio, 2021) Expansión. Consultado de https://expansion.mx/empresas/2021/06/29/sargazo-recuperacion-turismo-caribe-cancun
- (s.a) (s.f) matplotlib.pyplot.plot Documentacion de Matplotlib. Consultado de https://matplotlib.org/stable/api/as7gen/matplotlib.pyplot.plot.html
- (s.a) (s.f) matplotlib.contour.QuadContourSet Documentacion de Matplotlib. Consultado de https://matplotlib.org/stable/api/contourapi.html#matplotlib.contour.QuadContourSet
- (s.a) (s.f) matplotlib.image.AxesImage Documentacion de Matplotlib. Consultado de https://matplotlib.org/stable/api/imageapi.html#matplotlib.image.AxesImage
- (s.a) (s.f) matplotlib.pyplot.imshow Documentacion de Matplotlib. Consultado de https://matplotlib.org/stable/api/asgen/matplotlib.pyplot.imshow.html
- (s.a) (s.f) matplotlib.patches.Patch. Documentacion de Matplotlib. Consultado de https://matplotlib.org/stable/api/asgen/matplotlib.patches.Patch.html
- (s.a) (s.f) Shapely and geometric objects. Consultado de https://automating-gisprocesses.github.io/site/notebooks/L1/geometric-objects.html
- (s.a) (s.f) matplotlib.path. Documentacion de Matplotlib. Consultado de https://matplotlib.org/stable/api/pathapi.html

- (s.a) (s.f) matplotlib.pyplot.plot. Documentacion de Matplotlib. Consultado de https://matplotlib.org/stable/api/asgen/matplotlib.pyplot.plot.html
- (s.a) (s.f) Image Resolution and DPI. Consultado de https://largeprinting.com/resources/image-resolution-anddpi.html
- (s.a)(26 de dic, 2020) Apply a Gauss filter to an image with Python. Geeks for Geeks. Consultado de https://www.geeksforgeeks.org/apply-a-gauss-filter-to-animage-with-python/
- (s.a) (14 de julio, 2019) Python PIL GaussianBlur() method. Geeks for Geeks. Consultado de https://www.geeksforgeeks.org/python-pil-gaussianblurmethod/
- Banterla, D. (s/f) Texturas. Fac. Informatica San Sebastian. Consultado de http://www.ehu.eus/ccwintco/uploads/d/d7/Texturas.pdf
- Dabbura, I. (17 de spetiembre, 2018) K-means Clustering: Algorithm, Applications, Evaluation Methods,and Drawbacks. Towards data science. Consultado de https://towardsdatascience.com/k-means-clusteringalgorithm-applications-evaluation-methods-and-drawbacksaa03e644b48a
- gene (13 de abril, 2017) Geopandas Polygon to matplotlib patches Polygon conversion Stack Exchange. Consultado de https://gis.stackexchange.com/questions/197945/geopandaspolygon-to-matplotlib-patches-polygon-conversion
- gene (4 de junio, 2014). Converting Matplotlib contour objects to Shapely objects. Stack Overflow. Consultado de https://gis.stackexchange.com/questions/99917/convertingmatplotlib-contour-objects-to-shapely-objects
- Ghandi, R. (5 de Mayo, 2018) Naive Bayes Classifier Towards Data Science. Consultado de https://towardsdatascience.com/naive-bayes-classifier81d512f50a7c
- Gillies, S.(27 de sep, 2020) The Shapely User Manual. Shapely. Consultado de https://shapely.readthedocs.io/en/stable/manual.html
- Hall-Beyer, M. (2017) GLCM Texture: A Tutorial v. 3.0. University of Calgary. Consultado de https://prism.ucalgary.ca/bitstream/handle/1880/51900/texture%20tutorial%20v%2030%20180206.pdf?sequence=11&isAllowed=y
- jodag. (6 de mayo, 2020) Matplotlib - unable to save image in same resolution as original image. Stack Overflow. Consultado de https://stackoverflow.com/questions/34768717/matplotlibunable-to-save-image-in-same-resolution-as-originalimage34769840
- Korstanje, J. (7 de abril, 2021) The k-Nearest Neighbors (kNN) Algorithm in Python. RealPython. Consultado en https://realpython.com/knn-python/
- Lin, W. et al. (2010) Image Segmentation Using the Kmeans Algorithm for Texture Features. World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering
- Navlani, A. (2 de agosto, 2018) KNN Classification using Scikit-learn. Datacamp. Consultado en https://www.datacamp.com/community/tutorials/k-nearestneighbor-classification-scikit-learn
- R, Kirsten et al. (5 se septiembre, 2019) Performance of two multiscale texture algorithms in classifying silver gelatine paper via k-nearest neighbors. Open Archive Toulouse Archive Ouverte. Consultado de https://hal.archives-ouvertes.fr/hal02279362/document
- Rosebrock, A. (8 de agosto, 2016) k-NN classifier for image classification. pyImageSearch. Consultado en https://www.pyimagesearch.com/2016/08/08/k-nn-classifierfor-image-classification/
- Rosebrock, A. (28 de julio, 2014) A slic superpixel tutorial using python. pyImageSearch. Consultado en https://www.pyimagesearch.com/2014/07/28/a-slic-superpixel-tutorial-using-python/
- Rosebrock, A. (29 de diciembre, 2014) Accessing individual superpixel segmentations python. Consultado en https://www.pyimagesearch.com/2014/12/29/accessing-individual-superpixel-segmentations-python/
- tom10 (23 de Marzo, 2015) Python - convert contours to image. Stack Overflow. Consultado de https://stackoverflow.com/questions/29213238/pythonconvert-contours-to-image2921417
- A. Rosebrock, "k-NN classifier for image classification - PyImageSearch", PyImageSearch, 2021. [Online]. Available: https://www.pyimagesearch.com/2016/08/08/k-nn-classifier-for-image-classification/. [Accessed: 26- Jul- 2021].
- scikit-learn, "sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.24.2 documentation", Scikit-learn.org. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.fit. [Accessed: 26- Jul- 2021].
- scikit-image, "Module: segmentation — skimage v0.19.0.dev0 docs", Scikit-image.org, 2021. [Online]. Available: https://scikit-image.org/docs/dev/api/skimage.segmentation.html. [Accessed: 29- Jul- 2021].
- scikit-learn, "sklearn.cluster.KMeans — scikit-learn 0.24.2 documentation", Scikit-learn.org, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. [Accessed: 29- Jul- 2021].