

Proyecto final: Análisis de sargazo

Reconocimiento de Patrones - 0757

Facultad de Ingeniería,

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Martínez Ostoa N.I.
#315618648
Ing. en Computación
Facultad de Ingeniería, UNAM
Ciudad de México, México
nestor.martinez@iimas.unam.mx

Ramírez Bondi J. A.
#314634825
Ing. en Computación
Facultad de Ingeniería, UNAM
Ciudad de México, México
alejandrobondi@me.com

Profesores:
Dr. Boris Escalante Ramírez
Dra. Jimena Olveres Montiel
I.I.M.A.S. - U.N.A.M.

Resumen—

I. OBJETIVO

- Realizar un análisis de imágenes tomadas en distintos momentos de una playa en Puerto Morelos, Quintana Roo para detectar la presencia de sargazo en el océano y ayudar en la estimación del arribo de este elemento que afecta al turismo y demás actividades económicas.
- A partir del análisis, realizar alguna de las siguientes actividades concretas:
 - Detección del sargazo
 - Segmentación de las manchas del sargazo
 - Seguimiento del movimiento del sargazo
 - Cuantificación, en píxeles, del sargazo.

II. INTRODUCCIÓN

La llegada atípica de sargazo a las playas mexicanas y el Caribe ha provocado afectaciones económicas significativas en las entidades cuya economía está mayoritariamente basada en los ingresos por actividades turísticas.

Desde que se comenzó a contar con tecnología satelital, el monitoreo de las costas, corrientes y estado del tiempo se realizaba con imágenes obtenidas desde el espacio. Sin embargo, esta técnica de observación no ha sido muy eficaz para vigilar la llegada de sargazo a las playas mexicanas de manera precisa. Es decir, las imágenes satelitales nos permiten conocer si grandes masas de dicha alga van camino hacia el territorio mexicano, pero no nos permiten determinar con mayor precisión la playa a la que dicha biomasa llegará.

Lo anterior, nos ha llevado a pensar en una solución—mediante el análisis de imágenes—que pueda escalarse a una gran cantidad de playas, en especial aquellas con presencia turística, para mejorar la respuesta de las autoridades en el retiro y monitoreo del sargazo.

Este proyecto propone el desarrollo de modelos diversos para cuantificar, detectar y dar seguimiento a los cuerpos de sargazo a una playa ubicada en Puerto Morelos, Quintana Roo. El trabajo inicial, consistió en la recolección de imágenes con vista a la playa y el mar—por donde habitualmente llegan grandes cantidades de sargazo—por el equipo estacionado en una sede de la UNAM.

A partir de la secuencia de imágenes se realizó el procesamiento pertinente sobre las mismas, se segmentaron mediante diversas técnicas y se etiquetó por clases cada uno de los segmentos obtenidos. Finalmente, con la información antes mencionada, se entrenaron los modelos que se han considerado pertinentes para poder detectar si una imagen cuenta con la presencia de sargazo o no.

Así, se espera que este tipo de acercamientos para resolver la serie de problemas que se suscitan con la presencia del sargazo en las playas y mares del territorio mexicano, puedan ser atacados mediante el uso de imágenes obtenidas en puntos estratégicos distribuidos a lo largo de la zona costera afectada. Además, esta técnica ayudaría a tener información en tiempo real y dar respuesta lo antes posible—siendo fundamental la celeridad puesto que el sargazo entra en estado de descomposición y emite un olor muy desagradable para las personas ubicadas sobre la playa.

III. DESARROLLO

Para resolver los objetivos planteados en este proyecto diseñamos una arquitectura que consta de las siguientes fases:

1. Muestreo de imágenes
2. División de imágenes en imágenes de entrenamiento y prueba
3. Clasificación de imágenes
4. Extracción de características relevantes
5. Entrenamiento y evaluación de modelos

III-A. Muestreo de imágenes

Del *pool* original de imágenes, decidimos seleccionar las imágenes más representativas para identificar el sargazo. Nuestra base de imágenes constó de 11 imágenes:

- 1622134320.Thu.May.27_16_52_00.GMT.2021
- 1622134380.Thu.May.27_16_53_00.GMT.2021
- 1622134620.Thu.May.27_16_57_00.GMT.2021
- 1622138440.Thu.May.27_18_00_40.GMT.2021
- 1622144920.Thu.May.27_19_48_40.GMT.2021
- 1622157720.Thu.May.27_23_22_00.GMT.2021
- 1622732400.Thu.Jun.03_15_00_00.GMT.2021
- 1622732880.Thu.Jun.03_15_08_00.GMT.2021
- 1623441780.Fri.Jun.11_20_03_00.GMT.2021
- 1623442440.Fri.Jun.11_20_14_00.GMT.2021
- 1623442740.Fri.Jun.11_20_19_00.GMT.2021

A continuación se muestran dos de estas imágenes:



Figura 1: 1622138440.Thu.May.27_18_00_40.GMT.2021



Figura 2: 1622134380.Thu.May.27_16_53_00.GMT.2021

III-B. División de imágenes en imágenes de entrenamiento y prueba

Para esta sección, lo que hicimos fue lo siguiente:

1. Rescalamiento de las imágenes originales de 1280×960 a imágenes de 900×900
2. División en ventanas de 100×100
3. División en imágenes de entrenamiento y prueba

En la figura ?? se muestran varias imágenes de 100×100 para las clases sargazo, cielo, agua, costa y sargazo muerto respectivamente.



Figura 3: Imágenes de 100×100 pixeles para las clases sargazo, cielo, agua, costa y sargazo muerto respectivamente

III-C. Clasificación de imágenes

Una vez que contamos con el conjunto de entrenamiento y prueba, realizamos una clasificación supervisada de las imágenes de 100×100 pixeles en 5 clases:

1. Sargazo
2. Sargazo muerto
3. Agua
4. Cielo
5. Costa

Después de realizar esta clasificación, nos quedó la siguiente división:

	Entrenamiento	Prueba
Sargazo	134	53
Sargazo muerto	128	45
Agua	151	57
Cielo	78	29
Costa	309	116
Total	800	300

Cuadro I: Cantidad de imágenes de entrenamiento y prueba divididas en las 5 clases

III-D. Extracción de características relevantes

Para la extracción de características, lo que hicimos fue hacer un análisis de texturas con la matriz de co-ocurrencia y las características de Haralick. Las características que seleccionamos para nuestro vector de pruebas fueron las siguientes:

- Contraste
- Disimilaridad
- Homogeneidad
- Energía
- Correlación
- ASM

III-E. Entrenamiento y evaluación de modelos

Para el entrenamiento y evaluación de modelos, probamos con los siguientes clasificadores:

- k -vecinos
- Máquina de Soporte Vectorial con kernel *rbf*
- Clasificador Gausiano de Proceso

- Árboles de decisión
- *Random Forest*
- Perceptrón multi capa
- Clasificador *Ada Boost*
- Clasificador Bayesiano
- Análisis por discriminante cuadrático

A continuación se muestran los resultados (precisión) de cada clasificador:

	classifier	score
4	Random Forest	0.730000
3	Decision Tree	0.636667
6	AdaBoost	0.616667
8	QDA	0.616667
2	Gaussian Process	0.603333
5	Neural Network	0.596667
0	Nearest Neighbors	0.573333
1	SVM	0.543333
7	Naive Bayes	0.493333

Figura 4: Reporte de precisión para cada clasificador

Podemos observar que el que mejor desempeño tuvo fue el clasificador de árboles aleatorios. Este criterio (precisión) lo tomamos como un valor discriminante para analizar la precisión para la clase **sargassum** pues el valor mostrado en la figura ?? es el promedio general para las 5 clases. Siguiendo esta lógica, obtuvimos los siguientes resultados:

- *Random Forest*:

	precision	recall	f1-score	support
coast	0.87	0.91	0.89	116
dead_sargassum	0.62	0.62	0.62	45
sargassum	0.56	0.34	0.42	53
sky	0.90	0.90	0.90	29
water	0.53	0.68	0.60	57
accuracy			0.72	300
macro avg	0.70	0.69	0.69	300
weighted avg	0.72	0.72	0.71	300

Figura 5: Reporte de clasificación para *Random Forest*

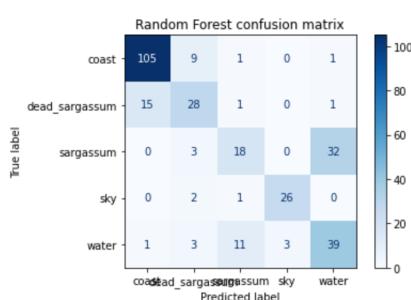


Figura 6: Matriz de confusión para *Random Forest*

- *Decision Tree*:

	precision	recall	f1-score	support
coast	0.76	0.96	0.84	116
dead_sargassum	0.47	0.18	0.26	45
sargassum	0.30	0.13	0.18	53
sky	0.84	0.72	0.78	29
water	0.51	0.79	0.62	57
accuracy			0.64	300
macro avg	0.58	0.56	0.54	300
weighted avg	0.59	0.64	0.59	300

Figura 7: Reporte de clasificación para *Decision Tree*

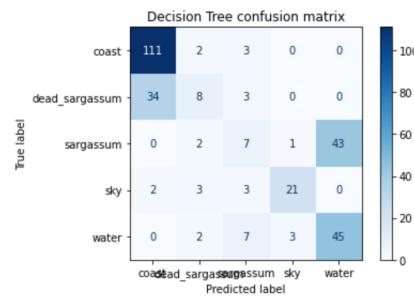


Figura 8: Matriz de confusión para *Decision Tree*

Podemos observar que *Random Forest* a pesar de obtener una precisión de 0,72, la precisión para el sargazo es de apenas 0,56. Mientras que para *Decision Tree*, a pesar de obtener una precisión de 0,64, la precisión de clasificación del sargazo es de apenas 0,3.

IV. RESULTADOS

V. CONCLUSIONES

Al revisar los resultados

VI. SECCIONES DE CÓDIGO RELEVANTE

VI-A. Extracción de características

```
def extract_haralick_features(grayscale_image, args):
    :
    distances = args['distances']
    angles = args['angles']
    glcm = greycomatrix(grayscale_image, distances,
                        angles)
    features = [
        np.mean(greycoprops(glcm, 'contrast')),
        np.mean(greycoprops(glcm, 'dissimilarity')),
        np.mean(greycoprops(glcm, 'homogeneity')),
        np.mean(greycoprops(glcm, 'energy')),
        np.mean(greycoprops(glcm, 'correlation')),
        np.mean(greycoprops(glcm, 'ASM'))
    ]
    return features

def label_encode(y):
    le = LabelEncoder()
    le.fit(y)
    return (le.transform(y), le.classes_)

def get_dir_contents(path):
    cts = os.listdir(path)
    return [c for c in cts if c[0] != '.']
```

Figura 9: Matriz de confusión para *Decision Tree*

- *Decision Tree*:

```

def get_X_y_vectors(path,
    feature_extraction_function, args):
X = []
y = []
for label in get_dir_contents(path):
    for file_name in get_dir_contents(path +
label + "/"):
        image = cv2.imread(path + label + "/" + 
file_name, cv2.IMREAD_GRAYSCALE)
        features = feature_extraction_function(
image, args)
        X.append(features)
        y.append(label)
y, classes_ = label_encode(y)
return (np.array(X), y, classes_)

#-----#
#      Feature extraction
#-----#
TRAIN_IMAGES_PATH = 'dataset/train/'
TEST_IMAGES_PATH = 'dataset/test/'

haralick_args = {
    'distances': [1],
    'angles': [0, np.pi/4, np.pi/2, 3*np.pi/4]
}

X_train, y_train, train_classes_ = get_X_y_vectors(
    TRAIN_IMAGES_PATH, extract_haralick_features,
    haralick_args
)
X_test, y_test, test_classes_ = get_X_y_vectors(
    TEST_IMAGES_PATH, extract_haralick_features,
    haralick_args
)

print(f"X_train: {X_train.shape}\ny_train: {y_train.
    shape}")
print(f"X_test: {X_test.shape}\ny_test: {y_test.
    shape}")
print(f"Train classes: {train_classes_}\nTest
    classes: {test_classes_}")

```

VI-B. Entrenamiento de clasificadores

```

#-----#
#      Model training and evaluation
#-----#
classifiers = [
    KNeighborsClassifier(n_neighbors=5, weights='
    distance'),
    SVC(),
    GaussianProcessClassifier(),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(),
    MLPClassifier(alpha=1, max_iter=1000),
    AdaBoostClassifier(random_state=0),
    GaussianNB(),
    QuadraticDiscriminantAnalysis()
]
names = ["Nearest Neighbors", "SVM", "Gaussian
    Process",
    "Decision Tree", "Random Forest", "Neural
    Network", "AdaBoost",
    "Naive Bayes", "QDA"]

scores = []

for name, classifier in zip(names, classifiers):
    classifier.fit(X_train, y_train)
    score = classifier.score(X_test, y_test)
    y_pred = classifier.predict(X_test)

    scores.append(score)

```

```

df = pd.DataFrame({'classifier': names, 'score': 
    scores})
df = df.sort_values(by='score', ascending=False)

```

VI-C. Evaluación de modelos

```

#-----#
#      Confusion matrix
#-----#
def evaluate(model, X_test, y_true, y_pred,
target_names, model_name):
    print(f"{model_name}")
    print(classification_report(y_true, y_pred,
target_names=target_names))
    plot_confusion_matrix(
        model, X_test, y_true, display_labels=
    target_names, cmap=plt.cm.Blues
    )
    plt.title(f"{model_name} confusion matrix")
    plt.show()

best_classifiers_df = df.iloc[:2]
for idx in np.array(best_classifiers_df.index):
    name = names[idx]
    clf = classifiers[idx]
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    evaluate(clf, X_test, y_test, y_pred,
    test_classes_, name)

```

REFERENCIAS

[1]