DESARROLLO DE SISTEMA DE ANÁLISIS DE VELOCIDADES DE FLUJO EN RÍOS A PARTIR DE IMÁGENES AÉREAS PARA SU UTILIZACIÓN EN SISTEMAS DE SOPORTE A LA DECISIÓN PARA MITIGACIÓN DE INUNDACIONES

Callejas, Manuel., López, Andrea., Rivera, César., Varela, Erick y Vidal, Noel.
Departamento de Electrónica e Informática, UCA, San Salvador, El Salvador
evarela@uca.edu.sv

Resumen—**La investigación expuesta en este artículo, propone el desarrollo de una aplicación para el cálculo de la velocidad promedio del caudal de un río, para el control de riesgos de inundación en zonas vulnerables, problemática que ha afectado a El Salvador durante los últimos años. Para la implementación de esta propuesta se ha desarrollado un programa en Python que utiliza el algoritmo YOLO, que permite identificar objetos dentro de frames de videos, así como el algoritmo de seguimiento por el método de centroides, que a través de la captura del punto central del objeto identificado almacena un registro de objetos reconocidos durante todo el video. Con este registro, se puede realizar el cálculo de la velocidad promedio de cada objeto identificado, utilizando métodos matemáticos establecidos, así como la conversión de pixeles a metros del resultado obtenido. Es posible desplegar toda esta información de manera clara y puntual en una interfaz gráfica, que permite al usuario final visualizar los resultados de los cálculos realizados a un video de entrada proporcionado.**

Palabras Clave – **Centroide, Darkflow, dataset, YOLO.**

I. INTRODUCCIÓN

Actualmente, el uso de machine learning se ha incrementado y propuesto como alternativa para aplicarse en áreas como predicciones y procesamiento de datos, monitoreo de cámaras, servicios, recomendación de productos, entre otros.

En El Salvador, este campo de la informática ha sido poco explotado, sin embargo, existe una diversidad de situaciones en las que puede aplicarse. Durante las últimas décadas, el país se ha enfrentado a una serie de catástrofes provocadas por fenómenos naturales, con influencia de actividades humanas que han desencadenado inundaciones, poniendo en riesgo a una parte de la población que se localiza en zonas rurales y urbanas vulnerables. Para dar seguimiento a estos problemas, el Ministerio de Medio Ambiente y Recursos Naturales (MARN) ha implementado los Sistemas de Alerta Temprana (SAT), que permiten controlar áreas de riesgo en base a una serie de parámetros de medición del nivel de agua en ríos; sin embargo, estos sistemas no cuentan con suficiente cobertura para la totalidad de zonas que se encuentran en riesgo, y por diversos factores, pueden llegar a representar un costo económico significativo para el Estado.

Como propuesta de solución, se ha planteado el desarrollo de una aplicación que permita la identificación y seguimiento de objetos en un video que capture el caudal de un río en un tiempo determinado, para así obtener las diferentes posiciones de los objetos para el cálculo de la velocidad a la que éstos se han desplazado con respecto al tiempo. Para ello, se utilizan algoritmos de reconocimiento y tracking de objetos, así como de procesamiento de datos, para obtener a través de métodos matemáticos la velocidad promedio del caudal del río.

II. METODOLOGÍA

A. Herramientas utilizadas:

- Python 3.6
- Anaconda 3.7
- Darkflow
- LabelImg
- Tkinter

B. Requerimientos del equipo:

- Sistema Operativo Windows 10
- Procesador Intel Core i5
- Memoria RAM de 8GB

C. Proceso de desarrollo:

1. Entrenamiento

Obtener un set de datos pre-entrenado que trabaje junto a YOLO, tal como COCO, o realizar un entrenamiento personalizado con los objetos que se deseen identificar, para ello se siguen los siguientes pasos:

1.1 Clonar el repositorio de Darkflow: https://github.com/thtrieu/darkflow, una vez creada una copia local, ubicarse en la carpeta del proyecto.

1.2 Obtener las imágenes a utilizar en formato .jpg y renombrarlas siguiendo un orden establecido y colocarlas en una carpeta dentro del proyecto Darkflow.

1.3 Etiquetar las imágenes a través de LabelImg, para así clasificar los objetos que se pretende reconocer. Esto genera un archivo .xml por cada imagen, que contiene el nombre del objeto y las coordenadas de éste dentro de la imagen.

1.4 Preparar el archivos de configuración que establece Darkflow (.cfg) y colocar el archivo de pesos (.weights) correspondiente a la versión de YOLO que se está utilizando.

Ejecutar el entrenamiento, con el siguiente comando:

python flow --model cfg/tiny-yolo-voc-1c.cfg --load bin/tiny-yolo-voc.weights --labels labels.txt --train --annotation C:\darkflow\annotations --dataset C:\darkflow\images --epoch 1500

Al terminar el entrenamiento, se obtienen dos archivos, de extensiones .meta y .pb, que representan al set entrenado listo para identificar objetos en imágenes.

2. Desarrollo de aplicación para identificación y seguimiento del objeto en el video

Crear un programa en Python, que permita cargar un video y obtener los resultados del análisis del video en cada uno de los frames que lo componen. Para realizar esta tarea, se debe importar Darkflow como si se tratase de una librería, y los archivos generados a través del entrenamiento ejecutado en el paso anterior.

En cada frame se obtienen los resultados, utilizando la función return_predict(), a la que se le coloca como parámetro de entrada el frame actual y devuelve como salida un diccionario de datos con todos los objetos identificados, junto a la clase a la que pertenecen y las coordenadas de inicio y fin del recuadro que rodea al objeto identificado. Estos resultados, se muestran en la Fig. 1, donde se puede visualizar la información en consola y el recuadro dibujado a través de la obtención de coordenadas del objeto.

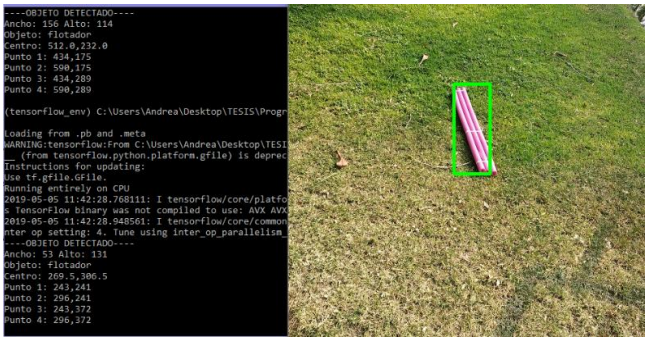

Fig. 1. Objeto identificado con YOLO e información en consola acerca de las coordenadas en las que se ubica.

Se debe dar seguimiento a cada uno de los objetos identificados en todos los frames del video, de manera que, aunque se identifique el mismo tipo de objeto, cada uno debe tener un id único que permita reconocerlo y saber que se está tratando con el mismo objeto, aunque éste se haya desplazado. Para ello, se utiliza el algoritmo de tracking por centroides, que consiste en calcular el punto central de cada objeto y darle seguimiento verificando la distancia mínima de un centro a otro en diferentes frames.

En la Fig. 2., se muestra un diagrama que representa los centroides de los objetos identificados.

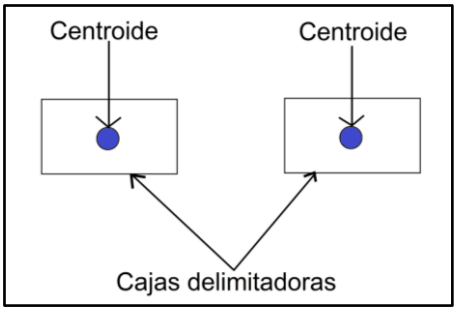

Fig. 2. Centroides de objetos identificados en imágenes.

Para la implementación de este algoritmo, se ha utilizado una librería de libre distribución, provista por el sitio en línea Pyimagesearch. Al realizar una instancia de la clase denominada CentroidTracker, se coloca como parámetro de entrada las coordenadas de inicio y fin del recuadro que rodea al objeto y que representa su identificación dentro del frame. El cálculo del centroide se hace internamente a través de esta librería, que también permite asignar un id, actualizar la información en caso de ser existente o eliminar un objeto después de cierto número de frames en los que ha desaparecido.

Además, se debe crear un diccionario de datos que contenga todos los objetos que han sido rastreados a lo largo

de la duración del video. Cada elemento que pertenece a este diccionario contiene la información del id del objeto identificado, el centroide en el frame actual y un arreglo con todos los centroides en los que se ha posicionado.

3. Cálculo de la velocidad promedio y conversión de coordenadas digitales a físicas

Luego de obtenerse todos los objetos identificados junto a la información de sus coordenadas en todos los frames, se debe procesar sus centroides para el cálculo de la velocidad promedio. Esto se realiza de la siguiente manera:

3.1 Se obtiene una velocidad para cada par de coordenadas, restando a cada componente, la de su instante anterior.

3.2 Al realizar la resta, se obtienen dos resultados, uno para la componente en X y otro para Y, estos valores se dividen entre un tiempo T, que es el tiempo transcurrido entre cada par de coordenadas. Cada componente se debe elevar al cuadrado. El cálculo del tiempo se realiza con el tiempo transcurrido de un frame a otro.

3.3 Posteriormente, se realiza el cálculo de la raíz cuadrada de la suma de los resultados anteriores. Este valor es la velocidad para un par de coordenadas. Este proceso se realiza de manera sucesiva entre cada par de coordenadas, obteniendo así, un conjunto de velocidades.

3.4 Cada valor de velocidad obtenido se guarda en una lista de velocidades finales, el proceso descrito entre los pasos 3.1 y 3.3 es iterativo, y se realiza tantas veces como coordenadas agrupadas en pares se haya evaluado. Al finalizar, se calcula un promedio de todas las velocidades obtenidas y este resultado es la velocidad de desplazamiento del objeto que se está rastreando.

Cuando se ha obtenido y elegido el mejor candidato para un valor de velocidad más preciso, se debe transformar este resultado de pixeles a metros. Este proceso se lleva a cabo obteniendo los ppi (pixels per inch) de la pantalla en la que se está trabajando, con la fórmula descrita en (1).

$$ppi = tamaño\ físico\ de\ pantalla\ /\ resolución\ pantalla \quad (1)$$

Finalmente, la conversión se realiza como se describe en (2).

$$valor\ velocidad\ final = velocidad\ /\ ppi \quad (2)$$

Una vez realizados los pasos descritos anteriormente y acoplados con la interfaz gráfica desarrollada con Tkinter, se procede a la ejecución del programa utilizando la consola de Anaconda, y activando el ambiente de Tensorflow, para su correcto funcionamiento.

III. RESULTADOS

Después de ejecutar el entrenamiento para identificar un objeto denominado "flotador", se obtuvo dos archivos de extensión .meta y .pb, que se utilizaron para cargarse en la aplicación realizada en Python, la que al analizar un video, genera otro video de salida con los objetos identificados, rodeados por un recuadro y con su centroide y id correspondientes. La Fig. 3 muestra la aplicación con un frame del video resultado.

Fig. 3. Frame de video con objeto identificado junto a su centroide marcado.

Al finalizar el proceso de cálculo, se muestra en pantalla el resultado final de la velocidad promedio del objeto que es detectado en más frames del video, con su información, como se observa en la Fig. 4.

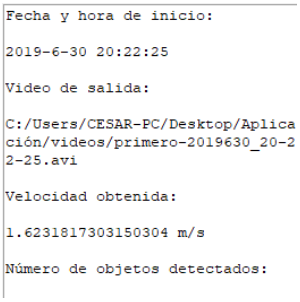

Fig. 4. Resultado de velocidad promedio desplegada en la aplicación.

IV. DISCUSIÓN

Los datos obtenidos en esta investigación, evidencian que el valor de velocidad resultante indica un promedio que varía de acuerdo a la cantidad de puntos identificados y procesados del objeto. Sin embargo, este valor cambia poco al realizar diferentes pruebas con el mismo video, por lo que se puede inducir que el valor de velocidad tiene precisión. Además, el video de prueba muestra un objeto

que se mantiene en movimiento sobre un eje central en el que gira con un diámetro aproximado de un metro, por lo que la trayectoria realizada, tomando el tamaño del objeto, la distancia recorrida y la duración en segundos del movimiento, indican que los valores se encuentran entre uno y dos metros sobre segundos. Aunque la aplicación desarrollada se probó con un video en movimiento sobre una superficie distinta a un río, se demuestra que se reconoce el objeto de estudio en las diferentes posiciones que ocupa, se calculan sus puntos, se realiza el seguimiento a estos puntos y se procesan para obtener un valor aproximado de velocidad promedio.

V. CONCLUSIONES

Por medio de la aplicación realizada, se contribuye a la investigación de temas poco abordados en el país, como machine learning, que puede tener un gran impacto en un futuro para la prevención de riesgos de inundación en zonas vulnerables. Además, se cumple con dos de los objetivos de desarrollo sostenible propuestos por la ONU en 2015, el primero corresponde a la innovación (objetivo 9) y el segundo está enfocado en lograr que los asentamientos humanos sean inclusivos, seguros y sostenibles (objetivo 11), propósitos que se han trazado con esta investigación.

Es posible realizar cálculos físicos a través del análisis y procesamiento de imágenes y videos, utilizando algoritmos de identificación y seguimiento de objetos, ubicando los pixeles que ocupan los objetos dentro de la imagen y transformando estas medidas virtuales a físicas, tomando como referencia la cantidad de pixeles por pulgada que cada pantalla soporta.

YOLO es uno de los algoritmos más rápidos y potentes para el reconocimiento de objetos, debido a que utiliza un análisis con una red neuronal una sola vez, sin necesidad de iterar muchas veces sobre la imagen o frame proporcionado. Para efectos de esta investigación, este algoritmo funcionó de manera óptima debido a que es bastante preciso, aún al utilizar una versión liviana. Además, se acopló con el algoritmo de seguimiento por el método de los centroides, que sólo necesitaba las coordenadas de salida proporcionadas por YOLO para obtener el centroide y darle seguimiento a este punto perteneciente al objeto, a lo largo del video.

REFERENCIAS

[1] O. Rincon. (Febrero 2019) “Instalando y entrenando YOLO en Windows10 con Darkflow”. [Online]. Available: https://medium.com/@oxanderv/instalando-y-entrenando-yolo-en-windows10-con-darkflow-6181166ee2ab?fbclid=IwAR2iWNrkplPhaBhScPIeVcOPmjpLBBvNd6m7fexWSwVKK3ZpMbgkQz4M3FQ

[2] "thtrieu/darkflow", *GitHub*, 2016. [Online]. Available: https://github.com/thtrieu/darkflow. [Accedido: 15- Jul- 2019].

[3] A. Rosebrock. (Noviembre 2018), “YOLO object detection with OpenCV”.[Online].Available:https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

[4] E. Gazar. (Agosto 2018) “Object Tracking with OpenCV”. [Online]. Available:https://ehsangazar.com/object-tracking-with-opencv-fd18ccdd7369

[5] “Tutorial de Python”. Accedido en Abril 2019. [Online]. Available: http://docs.python.org.ar/tutorial/3/real-index.html