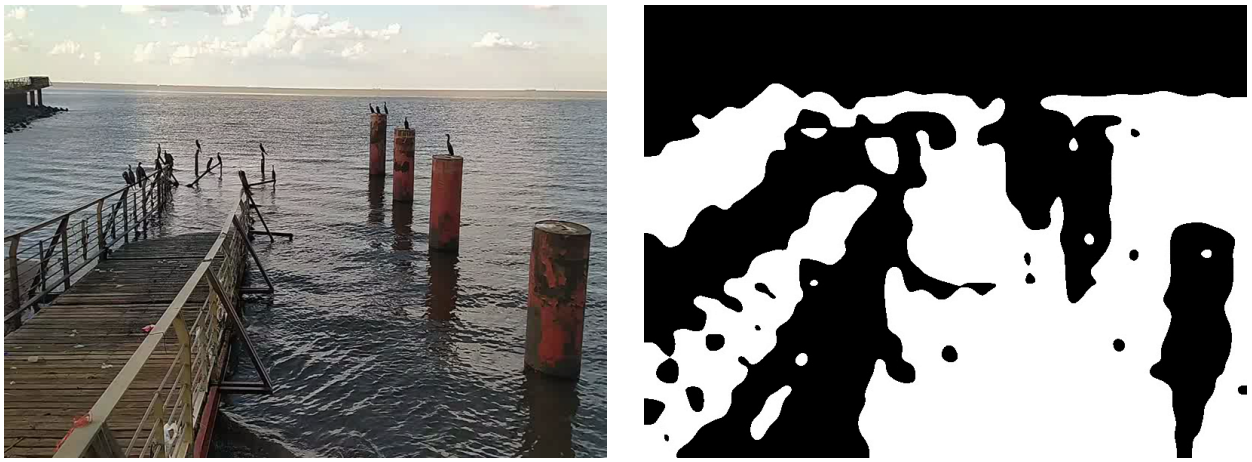
Informe del Trabajo Práctico Final

Belén Alvariñas, Constanza de Galvagni, Alan Ferrise, Leonel Saire Choque

Segmentación y clasificación de agua en videos

Contents



Segmentación de un video grabado en la Reserva Ecológica Ciudad Universitaria.

1 Introducción

1.1 Motivación y Objetivo

La detección automática de agua en videos tiene aplicaciones en áreas como la exploración interplanetaria, la eliminación de fondos dinámicos, la robótica y el análisis de videos aéreos. Sin embargo, este problema en específico recibió poca atención en la literatura. El objetivo de este trabajo es replicar y validar el método propuesto por *Mettes et al.* (2013), que utiliza un descriptor híbrido para caracterizar el comportamiento espacial y temporal que es propia del agua, y evaluar su efectividad en este tipo de conjunto de datos.

2 Dataset

El conjunto de datos utilizado consta de 260 videos distribuidos en dos clases: agua y no agua. Cada video contiene entre 750 y 1500 frames, con un tamaño de 800×600 píxeles. Tuvimos que explorar la web para encontrar el dataset, debido a que el link provisto por el paper no funciona. A continuación, se detallan las características principales del dataset:

- **Agua**: Incluye 160 videos con escenas de océanos, fuentes, lagos, ríos, canales y estanques.
- **No Agua**: Incluye 100 videos con escenas de árboles, fuego, banderas, nubes y vegetación.

3 Procedimiento

3.1 Preprocesamiento de Datos

- Primero se crea un nodo temporal. La idea del cálculo es obtener, para cada canal en cada pixel, el valor más frecuente que tiene durante la duración del video. Su expresión es la siguiente:

$$M(x,y)[c] = \max_{p} \left(\sum_{t=1}^{T} \mathbb{1}\{I_t(x,y)[c] = p\} \right) \tag{1}$$

  Donde $M(x,y)[c]$ es el nodo temporal y $c \in \{R, G, B\}$ representa los canales de color.

- Luego, a cada frame del video se le resta el nodo temporal y se obtiene el valor absoluto de la operación. Lo que se busca es resaltar las ondulaciones del agua, reduciendo los reflejos y colores predominantes.

```
def preprocess_videos(dataset):
    for video in dataset:
        temporal_node = obtain_temporal_node(video)
        for frame in video:
            residual_frame = |frame - temporal_node|
            save(residual_frame)
```

Listing 1: código para preprocesamiento

Figure 2: Nodo temporal

Figure 3: Frame del video residual

3.2 Generación de Descriptores

Dividimos a un frame de video en porciones de $n \times n$ que denominamos **parches**. Se busca representar cada parche teniendo en cuenta sus características espaciales y temporales. Para esto, se extrajeron descriptores espaciales y temporales:

- **Descriptor Temporal**: La idea es obtener una característica inherente del agua: su oleaje. Se tiene como hipótesis que el oleaje del agua es gradual y repetitivo, teniendo como consecuencia de que las olas ocurren en el mismo lugar a través del tiempo. Para lograr extraer este comportamiento, se obtiene el brillo medio de un parche en m frames consecutivos. A este resultado lo podemos interpretar como una señal, en donde en el eje y tenemos valores de intensidad y en el eje x cada uno de los m frames. Luego, aplicamos la Transformada de Fourier y le aplicamos la normal L1.

- **Descriptor Espacial**: Basado en histogramas de patrones binarios locales (LBP) para capturar información espacial de ondulaciones. Este analiza la textura comparando cada píxel con sus vecinos inmediatos y generando un histograma para capturar elementos visuales como ondas.

El siguiente código fue implementado para obtener los descriptores:

```
def obtenerDescriptores(dataset, tam_parche, num_frames):
    descriptores = []
    for video in dataset:
        for frame in video:
            temporal_descriptor = FourierTransform(mean_brightness(
               frame))
            spatial_descriptor = LocalBinaryPattern(frame, tam_parche
               )
            hybrid_descriptor = concatenate(temporal_descriptor,
               spatial_descriptor)
            descriptores.append(hybrid_descriptor)
    return descriptores
```

Listing 2: código para generación de descriptores

3.3 Clasificación y Segmentación

En este paso, se clasificó cada parche entre agua o no en función de los descriptores obtenidos en el paso anterior. Para esto, los mismos se clasificaron utilizando un bosque de decisión, es decir Random Forest, para estimar las probabilidades de cada parche local, que utilizamos luego para realizar la segmentación.

Segmentamos a través de umbralización: el parche es clasificado como agua sii su probabilidad supera 0.5.

El paper describe que se puede utilizar Markov Random Fields (MRF) previo a umbralizar para mejorar la segmentación y reducir los outliers, pero no lo implementamos para este trabajo por falta de tiempo.

4 Experimentación

Para generar los residuales tuvimos que "reparar" los videos. Ocurría que los videos residuales generados tenían el doble de velocidad y la mitad de duración, lo que afectaba al comportamiento temporal del agua. Al abrir un video, OpenCV exploraba la mitad de los frames que el encabezado del video decía que tenía. Creemos que la causa de esto era que los videos tenían frames duplicados que OpenCV ignoraba. Utilizamos un comando de ffmpeg para reparar todos los videos, logrando así el comportamiento deseado.

Para realizar el procedimiento descrito anteriormente, debemos definir algunas variables:

Tomamos $n = 20$ para el tamaño de los parches, que creemos que es un tamaño razonable en comparación a la resolución de los videos, que son de 800×600. Para los descriptores temporales, utilizamos $m = 200$ tal como lo utiliza el paper.

Con estos parámetros computamos los videos residuales de todos los videos del dataset.

Para el clasificador, establecimos el número de árboles en 100, un random state de 47 y la fórmula de entropía de Shannon como criterio de clasificación:

$$H(Q_m) = -\sum_k p_{mk} \log(p_{mk}) \tag{2}$$

Para el entrenamiento del Random Forest seguimos el procedimiento del paper con algunas variaciones: Dividimos el set de datos en 80/20. Extraemos aproximadamente 2500 parches para cada video de train. Luego, obtenemos los descriptores de cada uno: el descriptor temporal será de dimensión 200 y el temporal de dimensión 256. Combinamos ambos, obteniendo un vector de tamaño 456, que es el que utilizamos para entrenar. A este descriptor lo llamamos híbrido. En total, obtenemos 520000 decriptores para entrenar el modelo.

Entrenamos al Random Forest utilizando la GPU de Google Colab. No pudimos extraer los descriptores en la plataforma debido a que son intensivos en uso de CPU, lo que provoca que tarden mucho en computarse. En varias ocasiones el cómputo no finalizaba debido a que desconectaba el entorno de ejecución ya que nos excedíamos del límite de memoria, hasta que finalmente llegábamos al límite de tiempo que Colab permite. Finalmente, obtuvimos los descriptores en una computadora personal y los subimos a Colab para el entrenamiento.

Una vez entrenado el modelo, a cada video de test le extraemos todos los parches de sus 20 primeros frames, clasificando uno por uno. De ellos obtenemos una matriz de probabilidades de tamaño 40×30. Hacemos interpolación cúbica para incrementar su tamaño a 800×600 y discretizamos sus valores con umbralización para luego guardar el resultado como video.

Por último, grabamos algunos videos en Ciudad Universitaria y segmentamos sus 50 primeros frames para ver como respondía el clasificador con otros videos. En general, la segmentación es bastante certera, pero define como no agua a casi todo el video del estanque del Pabellón 0,

lo que no es correcto. Esto podría deberse a que el estanque no tiene un oleaje constante, sino que el agua está estancada.

5 Resultados

Definimos la calidad de la segmentación a través de la siguiente expresión:

$$S(V, M) = \frac{\sum_{i=1}^{|V|} s(V_i, M)}{|V|} \times 100\% \tag{3}$$

donde $|V|$ es la cantidad de frames del video, V_i el frame i, M la máscara correspondiente y a $s(V_i, M)$ se lo define como:

$$s(V_i, M) = 1 - \frac{\sum_{x=1}^{W} \sum_{y=1}^{H} |V_i[x, y] - M[x, y]|}{W \times H} \tag{4}$$

Debido a que los videos son estáticos, la segmentación ideal sería un video cuyos frames sean la máscara. Por lo tanto, para cada píxel de cada frame, se evalúa si coincide con el correspondiente pixel de la máscara. Mientras más parecido sea cada frame a la máscara, mejor será la segmentación.

Utilizando esa expresión, evaluamos cada video de test, obteniendo una calidad de segmentación promedio del 89.89%. La tabla 1 del paper establece que la calidad de segmentación utilizando descriptores híbridos y sin utilizar MRF es del $90.38\% \pm 0.5\%$. Si bien ese porcentaje es obtenido dividiendo al dataset en 60/40 y evaluando 250 frames de cada video, la calidad de segmentación que obtuvimos es muy similar.

(a) Frame (b) Segmentación obtenida (c) Máscara

Una caracterísitca a resaltar es que la segmentación obtenida suele tener outliers que van cambiando de lugar en cada frame, tal como podemos ver en la figura (b). Según el paper, utilizar MRF reduce estos outliers, logrando una segmentación más consistente a través del tiempo.

6 Conclusiones

El método replicado demostró ser efectivo para la identificación espacio-temporal de superficies de agua en videos, superando algoritmos tradicionales de reconocimiento de texturas dinámicas. Futuras investigaciones podrían abordar la implementación en tiempo real, así como también explorar las posibilidades de segmentar videos con cámaras en movimiento.