Tarea 2

CC5213 – Recuperación de Información Multimedia
Profesor: Juan Manuel Barrios
Fecha: 1 de octubre de 2020

El objetivo de esta tarea es implementar un detector de avisos comerciales en televisión. Dada una carpeta que contiene videos con la emisión de un canal de televisión (ver Figura 1) y una carpeta con avisos comerciales (ver Figura 2) usted deberá implementar un software que determine los segmentos de televisión donde fueron emitidos cada uno de los comerciales buscados.

Figura 1: Cuadros de ejemplo de videos de televisión.

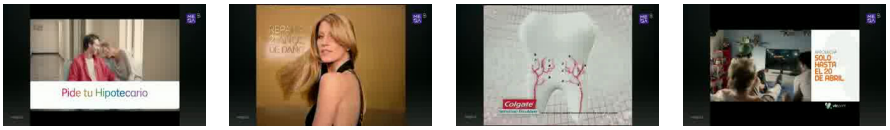

Figura 2: Cuadros de ejemplo de avisos comerciales.

El resultado de su tarea es un archivo de texto con todas las ocurrencias de los comerciales detectados en los videos de televisión. Cada ocurrencia debe incluir un valor de confianza donde un valor más alto significa que es más seguro que la detección sea correcta. El archivo debe tener formato de **cinco** columnas separadas por un tabulador (`\t`), cada aparición en una línea siguiendo el siguiente formato:

```
video_tv \t tiempo_inicio \t largo \t video_comercial \t confianza
```

Los tiempos se miden en segundos. Un posible archivo de salida es el siguiente:

televisión.mp4	87.3	30.1	comercial8.mp4	0.2943
televisión.mp4	420.8	15.6	comercial3.mp4	7.0121
televisión.mp4	1892.8	25.9	comercial2.mp4	5.0315
televisión.mp4	2087.5	30.1	comercial3.mp4	2.3875

La tarea debe recibir como parámetro una carpeta con los videos de televisión a procesar, una carpeta con los comerciales que se desea localizar y un nombre de archivo donde reportar los comerciales detectados. La tarea se debe dividir en los siguientes tres módulos.

Módulo 1: Extracción de características visuales

```
python tarea2-descriptores.py videos_tv/ descriptores_tv/
```

```
python tarea2-descriptores.py videos_comerciales/ descriptores_comerciales/
```

Lee todos los archivos con extensión .mp4 que están en la primera carpeta, calcula descriptores de contenido de cada video y escribe uno o más archivos en la segunda carpeta.

Hints para implementación: Una implementación simple consiste en seleccionar cuadros a una tasa constante (por ejemplo, uno por segundo), y para cada cuadro calcular un descriptor de contenido visual (similar a los de la tarea 1).

El conjunto de descriptores de cada video los puede guardar en formato binario junto con un archivo describiendo el origen de cada descriptor (video y tiempo).

Puede probar con distintos descriptores de contenido (grises, bordes, colores) y distintas formas de selección (dos por segundo, tres por segundo, uno por shot, etc.).

Módulo 2: Búsqueda por similitud

```
python tarea2-busqueda.py descriptores_tv/ descriptores_comerciales/ similares.txt
```

Lee los descriptores de los videos de televisión y comerciales y escribe en el archivo dado una lista con los cuadros más cercanos entre televisión y comerciales.

Hints para implementación: Llamaremos Q al conjunto de descriptores de los videos de televisión y R al conjunto de descriptores de los videos de comerciales.

Determine para cada descriptor q de Q el descriptor más cercano en R de acuerdo a alguna función de distancia (similar a la tarea 1).

Escriba en un archivo el nombre de cada cuadro de Q (video y tiempo), el cuadro de R más cercano (video y tiempo) y opcionalmente la distancia entre ellos.

Puede probar con distintos métodos de búsqueda o incluso calcular los k cuadros de R más cercanos (con $k \geq 1$).

Módulo 3: Detección de comerciales

```
python tarea2-deteccion.py similares.txt detecciones.txt
```

Lee la lista de cuadros similares del módulo anterior y detecta las secuencias provenientes de un mismo comercial y escribe en el archivo de salida todas las ocurrencias de los videos de comerciales junto con un valor de confianza.

Hints para implementación: Busque secuencias de vecinos más cercanos que provienen de un mismo comercial y que mantienen un mismo desfase de tiempo.

A continuación se muestra una lista de cuadros más cercanos de ejemplo, donde los descriptores se calcularon cada medio segundo:

Cuadro de televisión		Cuadro de comercial más cercano	
televisión.mp4	420.3	comercial5.mp4	23.8
televisión.mp4	420.8	comercial3.mp4	5.2
televisión.mp4	421.3	comercial3.mp4	5.7
televisión.mp4	421.8	comercial3.mp4	6.2
televisión.mp4	422.3	comercial7.mp4	12.9
televisión.mp4	422.8	comercial8.mp4	4.1
televisión.mp4	423.3	comercial3.mp4	7.7
televisión.mp4	423.8	comercial3.mp4	8.2
televisión.mp4	424.3	comercial7.mp4	12.9
televisión.mp4	424.8	comercial3.mp4	9.2

Al procesar esta lista de vecinos más cercanos se puede concluir que existe un segmento de al menos cuatro segundos de largo entre `televisión.mp4` (entre los segundos 420.8 y 424.8) y `comercial3.mp4` (entre los 5.2 y 9.2 segundos).

Notar que en ese segmento los tiempos de los cuadros similares entre ambos videos mantienen una diferencia constante de 415.6 segundos.

El valor de confianza de una detección podría ser, por ejemplo, el número de cuadros encontrados, el porcentaje del comercial encontrado, un valor relacionado al promedio de las distancias del más cercano u otro indicador que le parezca relevante.

Resultados de detección

Su tarea debe reportar la mayor cantidad de comerciales emitidos, con la mejor exactitud posible de los tiempos de inicio y largo, junto con la menor cantidad de detecciones incorrectas.

Junto con este enunciado encontrará conjuntos de prueba conteniendo videos de televisión, videos de comerciales y la respuesta con las detecciones esperadas.

Se publicará un programa llamado `tarea2-evaluar.py` que ejecuta los tres módulos de su tarea y evalúa el resultado de la detección. Cada detección reportada se considera correcta si intersecta una emisión real.

Su tarea será evaluada en los conjuntos de videos publicados y en otros conjuntos similares. **Su tarea no puede demorar más de 4 horas en total.**

Puede verificar visualmente cada aparición detectada usando *VideoLAN Player* con el siguiente comando[1]:

vlc "televisión.mp4" --start-time=[*segundos_inicio*] --run-time=[*segundos_largo*]

Entrega

Deberá entregar los códigos fuentes de su tarea y un archivo `Readme.txt` que explica brevemente su implementación, el o los descriptores usados, librerías usadas, forma de compilación y resultados obtenidos.

El plazo máximo de entrega es el **lunes 26 de octubre** hasta las 23:59 por U-Cursos. No se aceptarán tareas atrasadas.

La tarea puede ser implementada en **Python 3**, **C++ 11** o **Java 8** usando **OpenCV**. Puede utilizar cualquier librería gratuita (por ejemplo NumPy, SciPy u otra).

Debe enviar solo el código fuente de su tarea (archivos .cpp, .py o .java) y el archivo `Readme.txt`. No enviar videos ni descriptores calculados.

Evaluación

Se evaluarán tres aspectos: orden del código fuente (20%), claridad del archivo Readme.txt (20%) y la calidad de los resultados logrados (60%).

La calidad de los resultados logrados se medirá usando `tarea2-evaluar.py` sobre distintos conjuntos de videos.

La tarea es *individual*.

[1] Para más detalles usar el comando "`vlc --help`". Ver https://www.videolan.org/vlc/.