Tarea 2 - Comerciales

CC5204 - Búsqueda por contenido de Imágenes y Videos

Alumnos:	Patricio Isbej
	Elisa Kauffmann
Profesor:	Juan Manuel Barrios
Ayudante:	Sebastián Ferrada
Fecha:	2 de mayo de 2016

Índice

Índice de figuras

Índice de tablas

1. Introducción

El objetivo de esta tarea es comparar el uso de diferentes descriptores globales para resolver el problema de detección de spots publicitarios (avisos comerciales) en televisión. Se dispone de un archivo de video de cuatro horas de duración con la programación emitida por un canal de televisión local, junto con un total de 10 cortos de video, cada uno de largo entre 10 y 40 segundos, correspondientes a spots publicitarios emitidos por el canal durante ese período.

Se pide implementar un detector de comerciales que señale cuántas veces fue emitido cada comercial durante las 4 horas de programación, determinando exactamente el inicio y fin de cada aparición. Para implementar el detector de comerciales debe decidir un método de descripción del contenido (extracción de características), un método de búsqueda por similitud (comparación de descriptores), y un método de detección (encontrar secuencias parecidas).

Finalmente se debe señalar la empresa que realizó un mayor gasto en publicidad durante esas cuatro horas de programación (asumiendo valor constante de cada segundo al aire).

2. Resumen

Como solución se implementa en `Python` un detector de comerciales basado en descriptores de histogramas por zonas, y se realizan pruebas con distintas variantes del detector para determinar la configuración más efectiva y eficaz.

Los experimentos se hacen probando distintas frecuencias de muestreo de frames, cantidad de zonas por frame y número de bins de los histogramas. También se experimentó con distintos niveles de tolerancia durante la detección de secuencias parecidas.

Se concluye que el método usado reporta resultados de manera más eficaz usando una tolerancia más bien holgada en vez de estricta, independiente de la configuración. Usando una tolerancia del 80 %, de las configuraciones 100 % eficaces, la más eficiente en tiempo es aquella que usa una frecuencia de muestreo de

1 cada 20 frames, 1 zona e histogramas de 16 bins.

La empresa que realizó el mayor gasto en publicidad fue la que hizo los comerciales de Limon Soda, con un tiempo al aire de 128 segundos.

3. Diseño e implementación

Se implementa un detector de comerciales que señala cuántas veces fue emitido cada comercial durante las 4 horas de programación, determinando exactamente el inicio y fin de cada aparición. Para ello se divide el trabajo en 3 pasos:

1. Descripción del contenido (extracción de características)
2. Búsqueda por similitud (comparación de descriptores)
3. Detección (encontrar secuencias parecidas)

Se usa el lenguaje `Python` utilizando las librerías `OpenCV` para procesar imágenes y videos, `numpy` para manejo de vectores y matrices y `matplotlib` para graficar.

El reporte de detecciones se guarda en un archivo `.m3u` que por cada aparición de un comercial, contiene 3 líneas con el siguiente formato:

```
#EXTVLCOPT:start-time= [seconds]
#EXTVLCOPT:stop-time= [seconds]
../../base/mega-2014_04_25T22_00_07.mp4
```

donde la primera y la segunda línea indican el momento en que empieza y termina el comercial respectivamente (en segundos), y la tercera línea indica el video a reproducir, en este caso es el programa de 4 horas.

3.1. Descripción del contenido

A rasgos generales, en esta etapa se utiliza el descriptor de **histogramas por zonas** para caracterizar los frames de los videos guardando el descriptor de cada video en un archivo `.npy`, el cual será utilizado en el paso siguiente. Es importante mencionar que los frames de todos los videos son escalados al mismo tamaño (720×400 px) para normalizar y facilitar la división en zonas. Además, previo al

desarrollo de la tarea se verificó que todos los videos tuvieran el mismo *framerate* para mantener consistencia ($\sim$ 29.97 fps).

En esta etapa, los parámetros que varían en cada configuración son:

- frecuencia de muestreo de frames
- cantidad de zonas en las que se divide cada frame
- cantidad de bins de los histogramas

En detalle, el programa ejecuta los siguientes pasos:

1. Para cada archivo de video, se toma un frame cada cierta frecuencia, lo escala a un tamaño de 720×400 px y lo transforma a escala de grises.
2. Se divide cada uno de estos frames en la cantidad de zonas especificada por el tipo de configuración, y usando la función `calcHist` de `OpenCV` se obtiene un histograma por cada zona en forma de un arreglo.
3. Se crea el descriptor completo del frame: un arreglo que es la concatenación de los vectores de los histogramas obtenidos en el paso anterior.
4. Se crea el descriptor del video completo construyendo un arreglo `numpy` de todos los arreglos que describen cada frame del video.

El código se encuentra en el archivo `describe.py`

3.2. Búsqueda por similitud

Durante esta etapa, usando los descriptores creados anteriormente, por cada frame del video de 4 horas se encuentra el frame del comercial que más se le parece. Para ello se comparan todos los vectores descriptores de cada comercial contra cada frame del video de 4 horas, utilizando distancia Manhattan. Para facilitar el procesamiento se crea un diccionario que guarda, por cada comercial, su título, un número identificador y la cantidad de frames procesados. Un ejemplo de uno de los diccionarios creados se pueden observar en el cuadro 1.

Específicamente, el programa realiza los siguientes pasos:

Id	Nombre	Frames
0	mr big (2).npy	95
1	scotiabank.npy	76
2	ripley dias r.npy	63
3	sodimac homecenter.npy	62
4	donnasept.npy	61
5	dove desodorante hombre.npy	45
6	entel 4g corto.npy	81
7	johnson mundial.npy	60
8	limon soda.npy	96
9	bilz y pap.npy	105

Cuadro 1: Bosquejo del diccionario creado para la configuración de 1 descriptor cada 10 frames.

1. Se crea un diccionario que mapea los títulos de los comerciales a un identificador y a la cantidad de frames obtenidos en la etapa anterior.

2. Se cargan en memoria los descriptores de video de todos los 10 comerciales y usando el diccionario se construye una matriz de dimensiones 10×3. La matriz contiene, para cada comercial, una fila con su identificador, su título y su descriptor. (Ver cuadro 2).

3. Se carga el descriptor del programa de 4 horas y se compara cada frame contra todos los frames de los 10 comerciales. Para medir la similitud entre vectores descriptores se utiliza distancia Manhattan (utilizando la función `norm` de `OpenCV` con el parámetro `NORM_L1`). Así se va construyendo un vector de "calces" con tantas filas como frames del programa de 4 horas procesados. Cada fila contiene el índice del frame y el identificador del comercial que más se parece al frame del programa.

4. El vector de "calces" se guarda en un archivo `.npy` para ser usado en el paso siguiente

El código se encuentra en el archivo `compare.py`

com_id	nombre archivo	descriptor comercial
0	mr big (2).npy	[[],[],...,[],[]]
1	scotiabank.npy	[[],[],...,[],[]]
2	ripley dias r.npy	[[],[],...,[],[]]
3	sodimac homecenter.npy	[[],[],...,[],[]]
4	donnasept.npy	[[],[],...,[],[]]
5	dove desodorante hombre.npy	[[],[],...,[],[]]
6	entel 4g corto.npy	[[],[],...,[],[]]
7	johnson mundial.npy	[[],[],...,[],[]]
8	limon soda.npy	[[],[],...,[],[]]
9	bilz y pap.npy	[[],[],...,[],[]]

Cuadro 2: Bosquejo de la matriz de identificadores y descriptores de los comerciales

3.3. Detección

En esta etapa se detecta la aparición de los comerciales en el video de 4 horas haciendo uso del diccionario y el vector de "calces" creados en las etapas anteriores. La idea general es usar máscaras, una por cada comercial, para ir recorriendo el vector de "calces" y comparar la similitud entre segmentos de éste con las máscaras usando distancia de Hamming. La idea se ilustra en la figura **??**.

La distancia de Hamming es calculada usando una función específica implementada para este caso en particular.

Detalladamente, la implementación lleva a cabo los siguientes pasos:

1. Se cargan el diccionario y el vector de "calces" en memoria a partir de los archivos generados en la etapa anterior.

2. Para cada comercial en el diccionario se crea una máscara: una matriz con tantas filas como la cantidad de frames del comercial y 2 columnas. En la primera columna se repite para todas las filas el identificador del comercial y en la segunda columna se enumeran todos sus frames. Un ejemplo de máscara se puede ver en la figura 3.

3. Para cada línea del vector de "calces" se alínea la máscara del comercial

indicado en la posición del frame indicado. Se calcula la similitud entre la máscara y la sección del vector de "calces" usando distancia de Hamming. Este proceso se ilustra en la figura 1

4. Se reporta una aparición de un comercial cuando la similitud calculada en el paso anterior está dentro de cierto rango de tolerancia. En tal caso no se sigue trabajando con las líneas que corresponden a los frames de comercial, y se coloca el puntero en la posición del último frame de la máscara en el vector de "calces".

5. Al encontrar un comercial, se guarda el número del frame inicial y final del calce y se calcula el tiempo de inicio y fin del comercial. Estos datos se guardan en un archivo `.m3u` para crear la lista de reproducción que reporta las detecciones.

id comercial	frame
2	0
2	1
2	2
2	3
2	4
2	5
2	6
2	7
2	8
2	9
2	10
2	11
2	12
2	13
2	14

Cuadro 3: Bosquejo de la máscara de un comercial creada para la configuración de 1 descriptor cada 30 frames

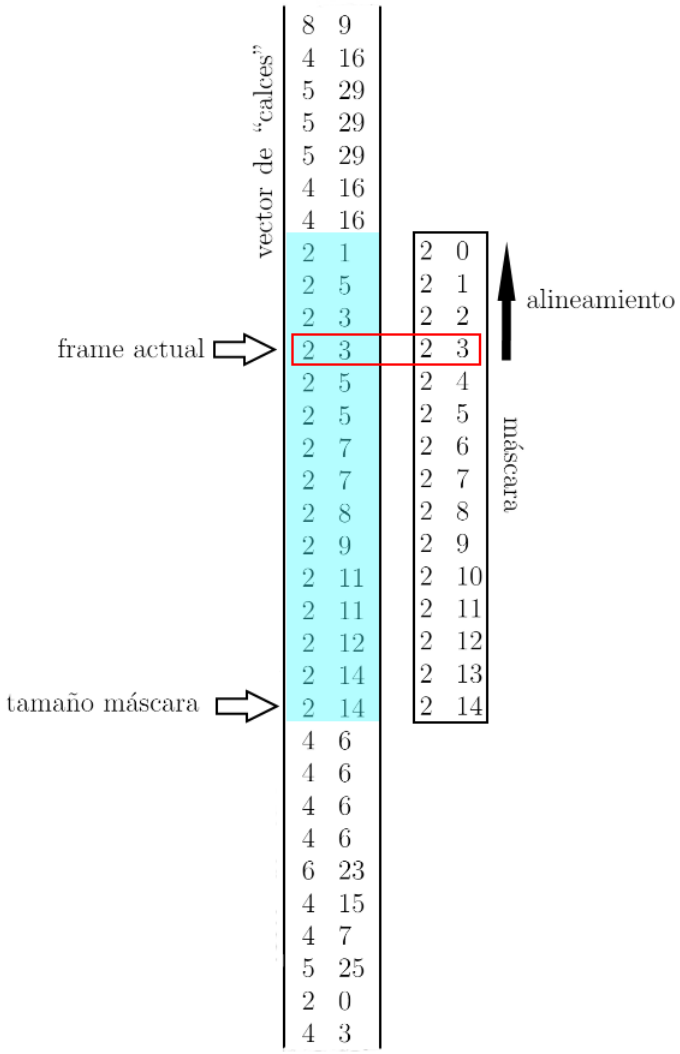

Figura 1: Esquema del proceso de detección de comerciales sobre el vector de "calces" usando máscaras.

4. Experimentos y resultados

Se realizan tests ejecutando `describe.py`, `compare.py` y `match.py` sucesivamente usando todas las combinaciones de los siguientes parámetros:

- Frecuencia de muestreo de frames: cada 30, 20 y 10 frames
- División por zonas: 1x1, 2x2 y 4x4
- Cantidad de bins: 16, 32 y 64

- Tolerancia: 80 % y 50 %

Para facilitar la ejecución, se crea el script `run_descriptors.sh` que para cada combinación ejecuta los 3 archivos y guarda los resultados y metadata de manera ordenada en distintos directorios.

Es importante destacar que la etapa 1, la de descripción es la que tomó más tiempo para todas las configuraciones. Las etapas de comparación y detección no tomaron más de 1 minuto en total para cada configuración.

Con respecto al tiempo al aire de cada comercial, se pueden observar los resultados en el cuadro 4. Se concluye que la empresa que realizó el mayor gasto en publicidad fue la que hizo los comerciales de Limon Soda, con un tiempo al aire de 128 segundos. Y la empresa que menos gasto realizó fue la del comercial de desodorante para hombre Dove, con 30 segundos al aire

Comercial	Cantidad de comerciales	Tiempo de emisión
limon soda	4	128
donnasept	4	81
bilz y pap	2	70
mr big (2)	2	63
johnson mundial	3	60
entel 4g corto	2	54
scotiabank	2	50
ripley dias r	2	42
sodimac homecenter	2	41
dove desodorante hombre	2	30

Cuadro 4: Tiempo al aire y cantidad de apariciones de cada comercial

La eficiencia de las distintas configuraciones se observa en la figura 2, mientras que la eficacia se puede ver en el cuadro 5. En este último se utiliza Valor-F.[1] para medir le eficacia. Un cuadro con el detalle de los falsos positivos y falsos negativos se puede observar en el cuadro 6 en la sección 6. Anexos

[1] `https://en.wikipedia.org/wiki/F1_score`

Con respecto a la eficiencia en espacio usado, la configuración más simple y más liviana: frecuencia=30 1×1 zonas y 16 bins, produjo un directorio de tamaño 1.8Mb; la mejor configuración produjo un directorio de 44Mb mientras que la configuración más compleja, frecuencia=10 4×4 zonas y 64 bins produjo un directorio de... 353Mb !

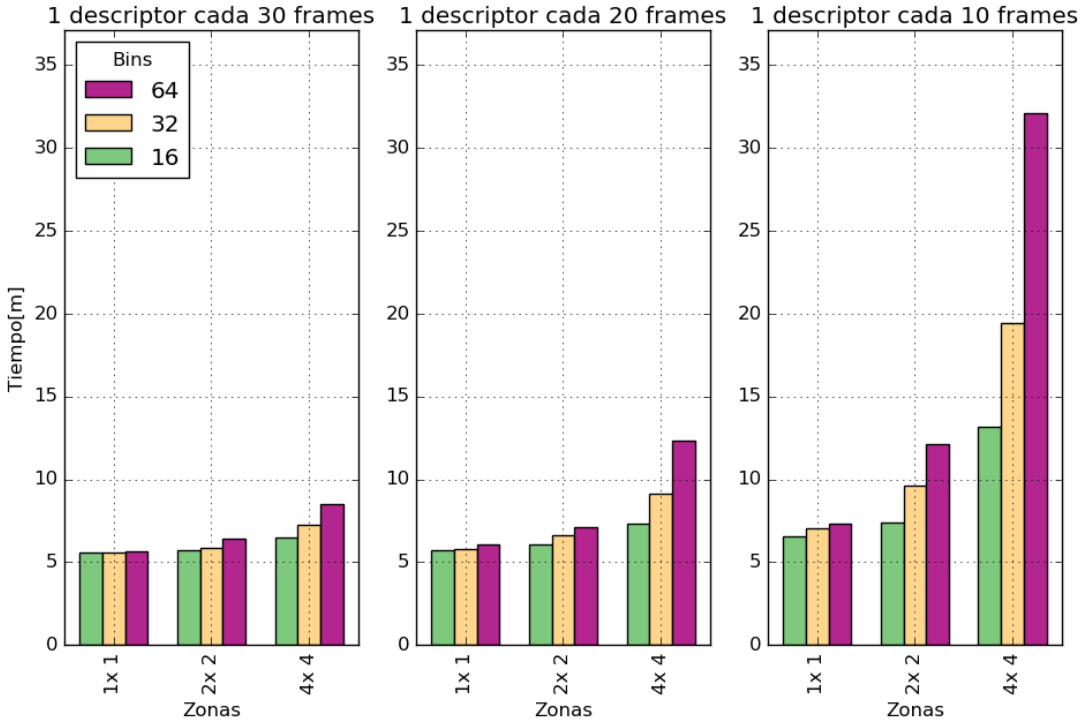

Figura 2: Eficiencia (en tiempo) de distintas configuraciones según frecuencia de sampleo, bins y número de zonas

	Frecuencia	30		20		10	
Division	Bins\Tolerancia	50 %	80 %	50 %	80 %	50 %	80 %
1x1	16	0.75	0.98	0.81	1	0.84	1
	32	0.75	1	0.78	1	0.89	1
	64	0.78	1	0.81	1	0.89	1
2x2	16	0.86	1	0.94	1	0.94	1
	32	0.89	1	0.86	1	0.94	1
	64	0.89	0.98	0.91	1	0.94	1
4x4	16	0.91	0.96	0.89	1	0.96	1
	32	0.91	0.96	0.86	1	0.94	1
	64	0.91	0.94	0.89	1	0.96	1

Cuadro 5: Eficacia de las distintas configuraciones según Valor-F

5. Análisis y conclusiones

De la figura 2, sobre la eficiencia, se puede notar que al usar una sola zona por frame, la diferencia introducida por el distinto número de bins utilizados no es muy notoria. Tampoco la diferencia es mucha para las distintas frecuencias de sampleo. Se pensaba que la frecuencia de muestreo de frames era lo que afectaría más la eficiencia del proceso, dado que el procesamiento de videos es lo que toma más tiempo, pero es más bien la cantidad de zonas y cantidad de bins lo que dispara los tiempos de procesamiento.

Sobre la eficacia, como se puede ver en el cuadro 5, claramente se concluye que una tolerancia *muy* estricta, como lo es un 50 %, genera resultados incorrectos. Con una tolerancia del 80 % los resultados son bastante razonables para cualquier configuración, es más, son 100 % eficaces para todas las configuraciones con frecuencia 10 y 20. Por lo mismo, con este método de detección, muestrear 1 frame de cada 10 es innecesario, independiente de los otros parámetros. Basta con muestrar 1 cada 20.

Es curioso que en muchas configuraciones, usar 32 bins reporta más falsos negativos que usar 16 o 64. Por ahora no se tiene una explicación para ello...

Sobre la cantidad de zonas, en general se puede decir que mientras más sean, más eficaz es el resultado de la detección. Pero para la frecuencia de muestreo de 1 frame cada 20 no se percibe una tendencia clara. Tampoco se tiene una explicación para ello por ahora.

6. Anexos

	Frecuencia	30		20		10	
Division	Bins\Tolerancia	50 %	80 %	50 %	80 %	50 %	80 %
1x1	16	15/0/10	24/0/1	17/0/8	25/0/0	18/0/7	25/0/0
	32	15/0/10	25/0/0	16/0/9	25/0/0	20/0/5	25/0/0
	64	16/0/9	25/0/0	20/0/5	25/0/0	20/0/5	25/0/0
2x2	16	19/0/6	25/0/0	22/0/3	25/0/0	22/0/3	25/0/0
	32	20/0/5	25/0/0	19/0/6	25/0/0	22/0/3	25/0/0
	64	20/0/5	25/1/0	21/0/4	25/0/0	22/0/3	25/0/0
4x4	16	21/0/4	25/2/0	20/0/5	25/0/0	23/0/2	25/0/0
	32	21/0/4	25/2/0	19/0/6	25/0/0	22/0/3	25/0/0
	64	21/0/4	25/3/0	20/0/5	25/0/0	23/0/2	25/0/0

Cuadro 6: Eficacia de las distintas pruebas
correcto/falsos positivos/falsos negativos