Grado Ingeniería Informática – ETSIIT - UGR

VISIÓN POR COMPUTADOR

Proyecto Final – Herramienta de recuperación de instancias mediante modelos de bolsas de palabras.

OSCAR JIMENEZ FERNANDEZ
23-12-2018

Tabla de contenido

1.- Definición del problema

1.1.- Descripción

El proyecto se centra en el desarrollo de una herramienta que haciendo uso de modelos de bolsa de palabras sea capaz de recuperar con la mayor eficacia posible las instancias de un objeto dado de consulta, sobre un conjunto de imágenes.

1.2.- Material y recursos empleados

En relación a los aspectos de implementación y experimentación para la obtención de resultados, emplearemos un conjunto de 5062 imágenes [1] sobre construcciones de Oxford proveídas por "Flickr" [9]. Adicional a las imágenes, encontramos una categorización de 4 etiquetas para cada una de las imágenes, en relación a la visibilidad del objeto en dicha imagen:

- Good: Una foto buena y clara del objeto o construcción.

- OK: Más del 25% del objeto es claramente visible.

- Bad: El objeto no se encuentra presente.

- Junk: Menos del 25% del objeto es visible, o hay un alto nivel de oclusión o distorsión.

También encontraremos un conjunto de descriptores SIFT calculados para cada una de las imágenes, aunque en nuestro caso, los computaremos empleando las funciones de OpenCV [3].

Por último, un fichero para cada imagen que contendrá los índices de las palabras ya asociadas a cada una de las imágenes, de entre un vocabulario de 1.000.000 de palabras, que nos resultará de gran utilidad a la hora de implementar nuestro modelo de recuperación.

En cuanto a aspectos software, emplearemos el IDE Spyder V-3.2.8 como entorno de desarrollo para Python V-3.6.5, junto con la librería de 'OpenCV'.

El desarrollo del proyecto será llevado a cabo en un ordenador portátil bajo el sistema operativo de Windows 10 con las siguientes características:

- Procesador: Intel® Core™ i7-4720HQ CPU @ 2.60GHz 2.60 GHz
- Memoria instalada (RAM): 8,00 GB
- Tipo de sistema: Sistema operativo de 64 bits, procesador x64

1.3.- Estudio del problema y definición de estrategias

Primero de todo, cabe diferenciar entre dos tipos de aplicaciones del reconocimiento de objetos. Estas son el reconocimiento de clases o categorías de un objeto, y el reconocimiento de instancias, que será de la que nos encargaremos en este proyecto. La técnica de reconocimiento se basa en buscar las coincidencias entre los descriptores calculados para nuestra imagen de consulta con los de todas las imágenes del dataset empleado. Con un proceso de verificación espacial, las coincidencias ambiguas son eliminadas al combinar objetos empleando una transformación afín global. Uno de los problemas de todo este proceso es el de

buscar las coincidencias de las características encontradas en toda la base de datos, que puede ser de gran tamaño, lo que provocará una pérdida de eficiencia en el proceso. Para reducir así la complejidad computacional de la búsqueda se emplearán aproximaciones como la técnica del vecino más cercano o la de hashing, que son las más comúnmente utilizadas.

Atendiendo a ello, la estrategia a seguir será la de obtención de los descriptores SIFT de cada una de las imágenes que usaremos del dataset, y empleando la técnica de clustering conocida como 'k-means', agruparemos estos descriptores en clusters, de los que obtendremos los centroides, que se corresponderán con cada una de las palabras visuales con las que construiremos nuestro modelo de bolsa de palabras para la recuperación de instancias. De este modo, una vez hemos extraído las palabras visuales, emplearemos la técnica de matching de vecino más cercano con ayuda de las funciones proporcionadas por OpenCV, para asociar los descriptores de cada imagen con aquellas palabras visuales más cercanas, lo que nos dará la bolsa de palabras asociada a cada una de estas imágenes. El tamaño del vocabulario será un factor importante en nuestro algoritmo, que determinará la calidad de los resultados como podremos observar en el análisis de los resultados.

Con la comparación de palabras visuales obtenemos un método más rápido que comparando directamente los descriptores de características. Hallaremos así el histograma de las palabras visuales asociadas a cada una de las imágenes, de manera que cuando pretendamos recuperar una instancia, calcularemos la intersección del histograma de consulta con el resto, empleándola como medida de similitud. Normalizaremos los histogramas y usaremos el coseno como medida de similitud entre dos histogramas.

Como comparación, emplearemos los archivos proporcionados en el enlace [2] a la página de descarga del dataset y demás recursos indicados en la sección 1.2, extrayendo los índices de las palabras de cada una de las imágenes y construyendo así el modelo empleando un vocabulario de 1.000.000 de palabras visuales, con el que realizaremos el mismo proceso de recuperación mediante la similitud de histogramas.

Además, implementaremos una mejora y desarrollaremos una serie de experimentos para comprobar cuál de las técnicas desarrolladas obtiene mejores resultados.

2.- Aspectos de implementación

Para este apartado, explicaremos el proceso seguido centrándonos en la implementación del código.

2.1.- Funciones y procesos

Lectura de imágenes

En primer lugar, procedemos a la lectura de las imágenes, para la cual hacemos uso de la librería 'walk' [4] para recorrer los ficheros (en este caso las imágenes) de una ruta dada, y junto con la función 'imread' de OpenCV [3], leer cada una de las imágenes e ir insertándolas en un vector.

Cálculo de descriptores

Procederemos ahora al cálculo de los descriptores SIFT para cada una de las imágenes cargadas, empleando las funciones 'SIFT_create' y 'detectAndCompute' proporcionadas en el módulo de OpenCV.

Clustering

Una vez calculados los descriptores en el paso anterior, obtendremos un modelo de palabras visuales generadas con la técnica de clustering empleando los descriptores asociados a las imágenes. En nuestro caso, emplearemos una que encontraremos en OpenCV, conocida como K-Means [8], a la que deberemos de pasar los descriptores como argumento, además de una serie de parámetros adicionales que deberemos definir.

El primero parámetro hace referencia al 'K', es decir, el número de clusters que pretendamos crear, que se corresponderá con el número de palabras visuales en los que agruparemos los descriptores, obteniendo un centroide para cada uno de esos clusters que será la palabra visual que lo representará. El segundo parámetro será el criterio de parada del algoritmo, y por último un tercero que influirá en la selección de los centroides para el cálculo de los clusters.

Como resultado de la aplicación de este proceso devolveremos los centroides obtenidos, que emplearemos en el cálculo de los histogramas de cada imagen.

Cálculo de bolsa de palabras

Para el cálculo de la bolsa de palabras podemos hacer uso de las funciones de matching 'BFMatcher' y 'knnMatch' o 'match' de OpenCV, en función de si queremos usar el método por fuerza bruta, o del vecino más cercano, pasando como argumentos los descriptores de la imagen y las palabras visuales obtenidas en el proceso anterior.

Cálculo de histogramas

Una vez tenemos la bolsa de palabras de una imagen, podemos calcular el histograma de esta, contando el número de repeticiones de cada palabra.

Cálculo de similitud entre histogramas

Cuando nos encontramos en la fase de recuperación de instancias, una vez tenemos el histograma de la imagen de consulta, así como los histogramas de cada una de las imágenes de la base de datos sobre las que ejecutar la consulta, procederemos al cálculo de la similitud de la imagen empleada de query, con el resto. Esta atiende a la siguiente fórmula:

$$sim(h_1, h_2) = \frac{h1 \cdot h2}{||h_1||\,||h_2||} = \cos\theta$$

De modo que, el mayor valor que obtendremos vendrá dado para cada histograma consigo mismo, el cual tendrá un valor máximo de 1.

Recuperación de instancias

Con la similitud entre la imagen de consulta y cada imagen del dataset, devolveremos aquellas instancias en función del orden decreciente para el valor de la similitud, pues como ya hemos comentado, aquellas cuyo valor de similitud sea más cercano a la unidad tendrán una mayor cantidad de palabras en común, lo que nos devolverá unos mejores resultados; aunque esto no tiene por qué ser del todo así, ya que instancias totalmente diferentes podrán darnos palabras en común, sin tener nada que ver una imagen con la otra en aspectos de recuperación de la propia instancia.

3.- Propuestas de mejora

Como aspectos de mejora cabría considerar, desde un punto de vista de la implementación de la herramienta, el alojamiento en un servidor de los histogramas previamente calculados, de manera que cuando se desee realizar una consulta, no sea necesario el cálculo de todos ellos, sino tan solo el de la imagen de consulta, y disparar el cálculo de la similitud para la recuperación. De cara a los aspectos de memoria de nuestro computador, esto es un problema puesto que como hemos comprobado en la experimentación, el cálculo de todos los histogramas cuando disponemos de un gran vocabulario y base de imágenes, requiere de un gran espacio, que, en mi caso, con 8GB de memoria RAM, tan solo daba con esfuerzo para el cálculo de unas 1000 imágenes. Como solución a ello, podemos implementar como mejora un modelo de índice invertido junto al de bolsa de palabras, de tal manera que no sea necesario el cálculo de todos los histogramas y similitud para la recuperación de instancias. En vez de ello consultaremos el número de ocurrencias de cada imagen en cada una de las palabras para realizar la recuperación.

Desde un punto de vista de mejora de los algoritmos, el empleo de diversas técnicas en el cálculo de los clusters, puede ser determinante a la hora de la obtención de resultados como hemos comprobado. Esto se pone igualmente de manifiesto en el paper de referencia [6], donde compara varias de estas técnicas; concretamente encontramos algunas como AKM con HAKM, consiguiendo unos mejores resultados con HAKM.

La verificación espacial es uno de los aspectos a tener en gran consideración, ya que esto puede mejorar notablemente la eficacia de los resultados obtenidos.

Como posible mejora del rendimiento de nuestra herramienta, implementaremos además un modelo de índice invertido con bolsa de palabras de manera que podamos reducir uno de los problemas que nos encontramos de memoria para el cálculo de todos los histogramas, y comprobar si conseguimos mejorar todo el proceso de consulta.

3.1.- Implementación de mejora

Índice Invertido

Desarrollaremos un modelo de índice invertido, de manera que tendremos una lista donde cada posición se corresponderá con el índice de una palabra, y cada posición dentro de cada índice de la lista se corresponderá con un índice a una imagen que contiene dicha palabra.

Query

En este caso, la query no la realizamos como anteriormente, empleando un histograma y calculando la similitud entre estos, sino que ahora, para cada palabra contenida en la imagen de consulta, comprobaremos en el índice invertido las imágenes que también están asociadas a las mismas palabras, de manera que contaremos las ocurrencias de cada imagen y devolveremos aquellas instancias que mayor número de ocurrencias tengan.

4.- Modo de uso de la herramienta desarrollada

4.1.- Herramienta

Para la experimentación y obtención de resultados, hemos empleado tanto la estrategia por la que extraemos nosotros mismos el vocabulario, como la que parte de las palabras asociadas a cada una de las imágenes del enlace dado [2], además de la que está basada en el índice invertido junto a la bolsa de palabras.

De cara al uso de la herramienta, emplearemos el vocabulario de 1M de palabras dado, ya que, de cara a los resultados, nos permitirá obtener una mayor eficacia y eficiencia en la recuperación de instancias.

Para su uso ejecutaremos la siguiente orden en un intérprete de Python:

$ RecuperarInstancia <índice de instancia a recuperar> <ruta de la carpeta con las imágenes sobre la que ejecutar la consulta> <ruta de palabras asociadas con las imágenes> <número de mejores instancias a devolver por la consulta>

Ejemplo: Situados en la carpeta "…/proyectofinal", donde encontramos las carpetas "imágenes" y "words", ejecutaríamos:

$ RecuperarInstancia 1 "imagenes/" "words/" 10 ; de modo que obtendremos las 10 mejores instancias para la imagen de nombre "img1.jpg" respecto de las imágenes contenidas en la carpeta "imágenes".

```
(base) C:\Users\OSCAR\Documents\MEGA\Ing. Informática\TERCERO\VC\proyecto final>python RecuperarInstancia.py 34 img_experimentos/exp2/ words/exp2/ 5
Iniciando lectura de imagenes...
Lectura de imágenes finalizada.

EXPERIMENTO TIPO 3:
Número de imágenes: 543
Inicio de lectura de palabras de imagenes...
Fin de lectura de palabras de imagenes.
Iniciando construcción de índice invertido...
Indice invertido finalizado.
Tiempo de construcción de indice invertido: 1.7277181148529053
Iniciando query...
Query terminada.
Tiempo de query: 0.011012077331542969
[4837.0, 34]
[842.0, 148]
[367.0, 118]
[301.0, 451]
[243.0, 494]
```

4.2.- Experimentación

Para realizar así toda la extracción de resultados, hemos definido un archivo 'main.py', que podrá ser ejecutado para comprobar la obtención de los resultados obtenidos en este documento.

Para ello deberemos de tener en cuenta la estructura de carpetas (también especificada en el archivo readme.txt), incluyendo en la carpeta 'img_experimentos/exp1/' y 'img_experimentos/exp2/' las imágenes que deseemos para cada uno de los experimentos. Además, en las carpetas 'words/exp1/' y 'words/exp2/', deberemos incluir los ficheros que incluyen las palabras para cada imagen, necesarias para el tipo de experimento 2 y 3.

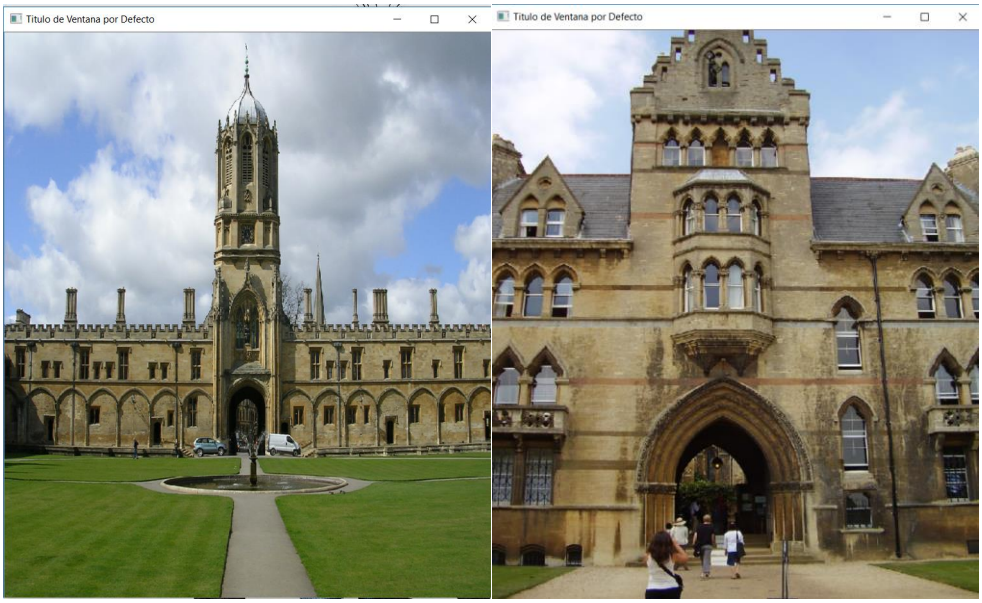
5.- Experimentación y resultados obtenidos

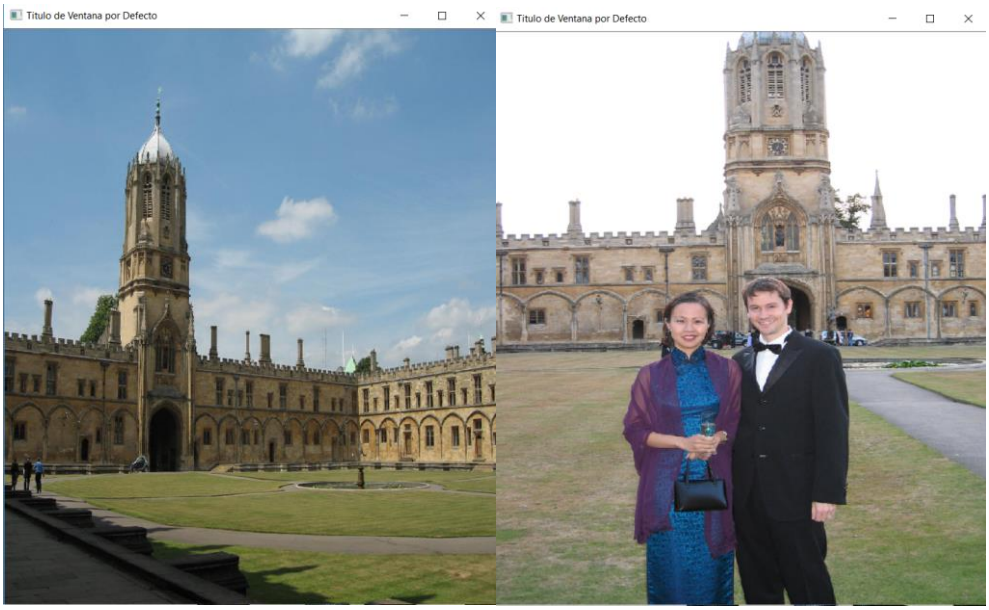
En el primero de los experimentos, extraemos nuestro vocabulario como hemos explicado en apartados anteriores, de un total de 210 imágenes, con un tamaño de vocabulario de 100 palabras.

A continuación, mostraremos los tiempos en las fases de construcción del diccionario y consulta en segundos, junto con las mejores 5 instancias recuperadas:

```
In [1]: runfile('C:/Users/OSCAR/Documents/MEGA/Ing. Informática/TERCERO/VC/proyecto final/main.py', wdir='C:/
Users/OSCAR/Documents/MEGA/Ing. Informática/TERCERO/VC/proyecto final')
Iniciando lectura de imagenes...
Lectura de imágenes finalizada.
Iniciando lectura de imagenes...
Lectura de imágenes finalizada.
Calculando descriptores...
Fin del cálculo de descriptores.

EXPERIMENTO TIPO 1:
Número de imágenes: 210
Inicio de clustering...
Fin de clustering.
Tiempo de construccion del vocabulario: 278.8059787750244
Inicio de calculo de histogramas...
Histogramas calculados.
Iniciando query...
Query terminada.
Tiempo de query: 11.285970687866211
```





El tiempo de construcción del diccionario es a tener en cuenta, aunque de cara a la implementación de una herramienta de este tipo, no sería relevante pues, como hemos indicado en aspectos de mejora, tan solo tendría que ser calculado una vez, alojándose en un servidor para su posterior consulta.

Respecto a la calidad de los resultados obtenidos, la primera imagen se corresponde con la imagen de consulta, que como era de esperar, es la primera en las instancias recuperadas. El resto de las instancias recuperadas podría mejorar, aunque no obtenemos resultados disparatados.

En este segundo caso, el vocabulario construido será de 1000 palabras.

```
EXPERIMENTO TIPO 1:
Número de imágenes: 210
Inicio de clustering...
Fin de clustering.
Tiempo de construccion del vocabulario: 2303.8896379470825
Inicio de calculo de histogramas...
Histogramas calculados.
Iniciando query...
Query terminada.
Tiempo de query: 10.34904408454895
```


Titulo de Ventana por Defecto


Titulo de Ventana por Defecto


Titulo de Ventana por Defecto


Titulo de Ventana por Defecto

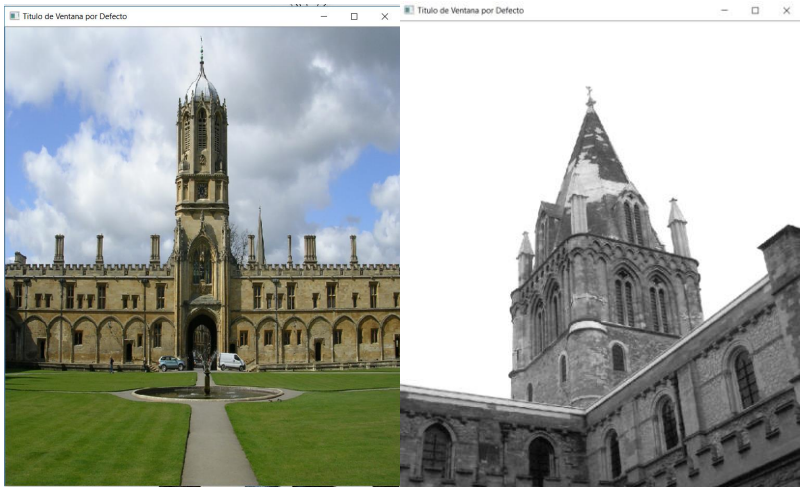

Titulo de Ventana por Defecto

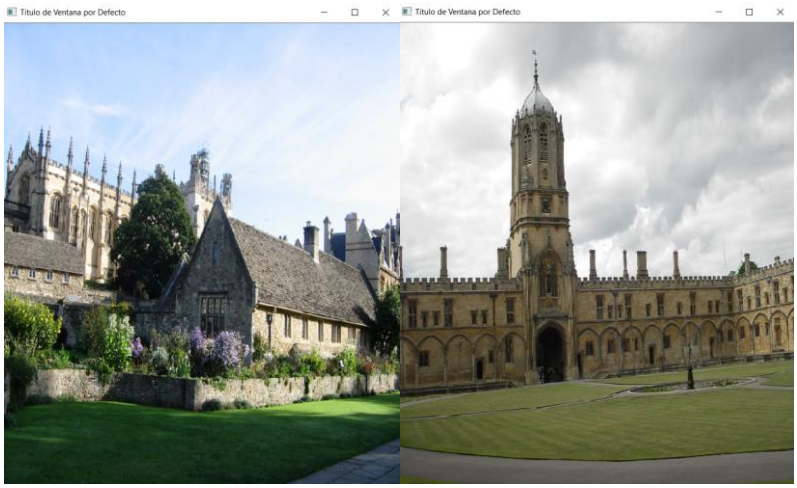
Podemos notar como el tiempo se multiplica casi por 10, al haber incrementado el vocabulario por 10 veces.

Los resultados sin embargo de las instancias recuperadas, no difieren en gran medida a simple vista de los anteriores obtenidos.

Ahora emplearemos el vocabulario de 1M de palabras, sobre la misma cantidad de imágenes:

```
EXPERIMENTO TIPO 2:
Número de imágenes: 210
Inicio de lectura de palabras de imagenes...
Fin de lectura de palabras de imagenes.
Inicio de calculo de histogramas...
Histogramas calculados.
Iniciando query...
Query terminada.
Tiempo de query: 9.91327714920044
```

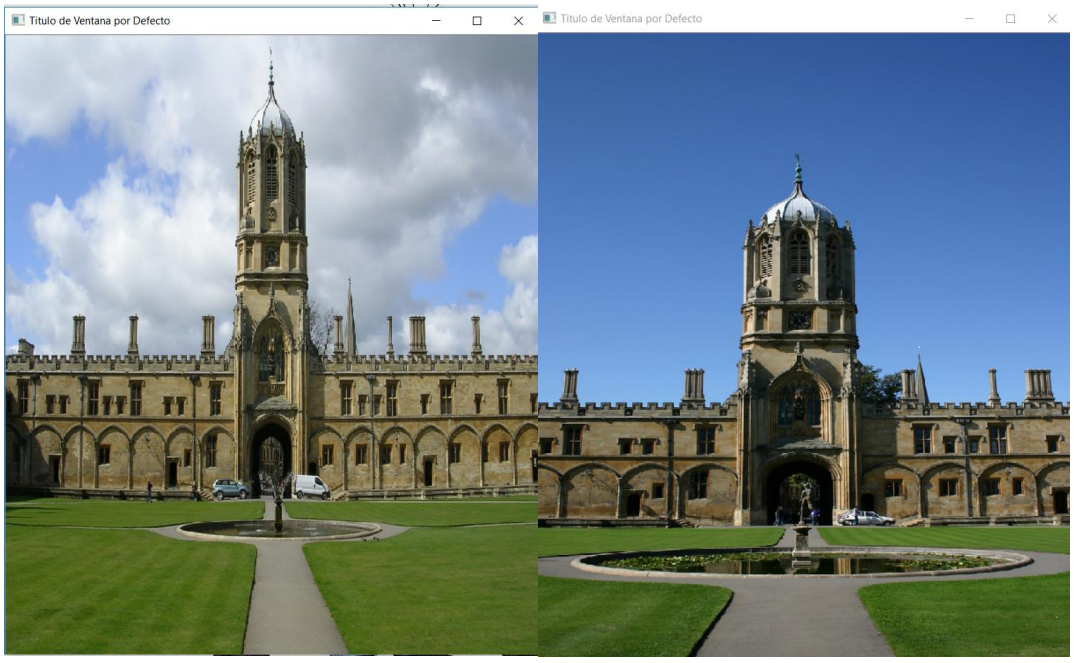


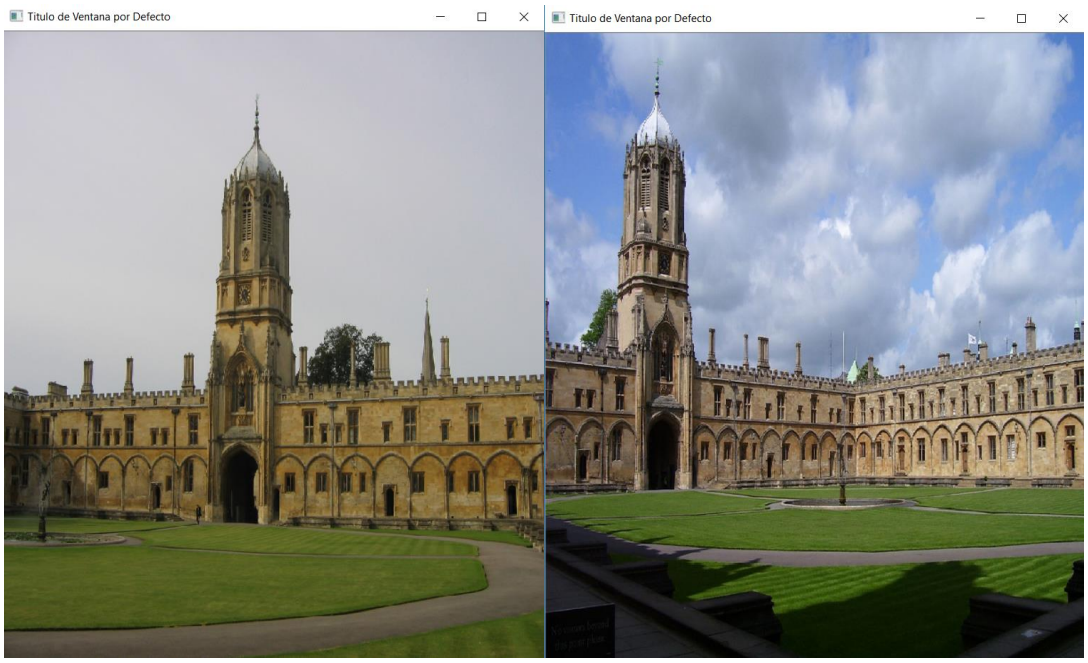


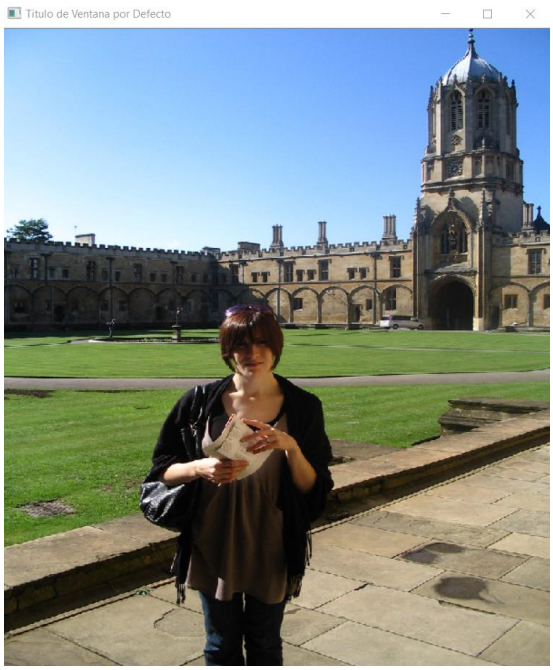
La ejecución en este caso es mucho más rápida al no tener que calcular el vocabulario, y a su vez el tiempo de consulta se ha reducido. Sin embargo, los resultados que hemos obtenido, distan bastante de ser los idóneos.

En este cuarto caso, ejecutaremos el experimento anterior, pero sobre toda la colección de imágenes del tipo que estábamos empleando (“church”).

```
EXPERIMENTO TIPO 2:
Número de imágenes: 543
Inicio de lectura de palabras de imagenes...
Fin de lectura de palabras de imagenes.
Inicio de calculo de histogramas...
Histogramas calculados.
Iniciando query...
Query terminada.
Tiempo de query: 25.777427673339844
```







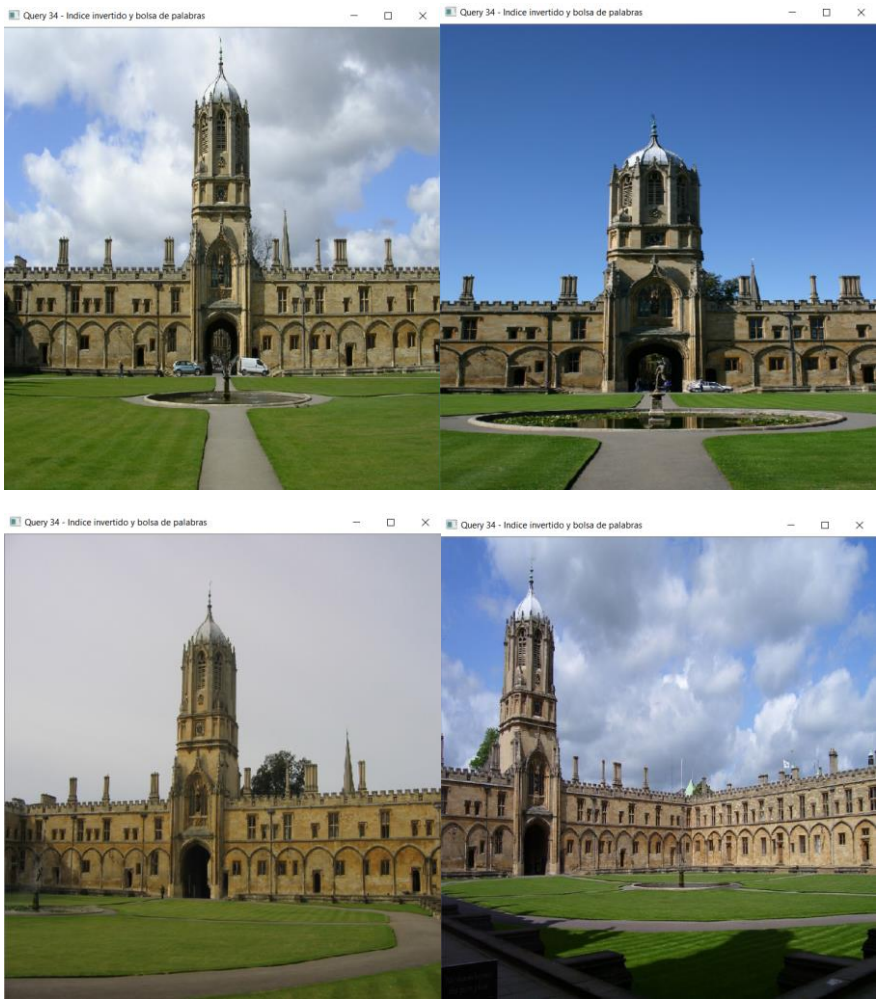
En esta ocasión, vemos como ahora sí los resultados son bastante buenos, aunque se nos presenta el problema de que, a mayor número de imágenes, mayor memoria es necesaria para el cálculo de histogramas, y a su vez, el tiempo de consulta aumenta al necesitar contrastar más imágenes.

Por último, vamos a comprobar si nuestra propuesta de mejora, implementando un modelo de índice invertido junto con la bolsa de palabras, nos proporcionará mejores resultados, tanto en eficiencia como en eficacia.

Tras la realización del experimento 3 obtenemos lo siguiente:

```
EXPERIMENTO TIPO 3:
Número de imágenes: 543
Inicio de lectura de palabras de imagenes...
Fin de lectura de palabras de imagenes.
Iniciando construcción de índice invertido...
Indice invertido finalizado.
Tiempo de construcción de indice invertido: 1.7819743156433105
Iniciando query...
Query terminada.
Tiempo de query: 0.008989810943603516
[4837.0, 34]
[842.0, 148]
[367.0, 118]
[301.0, 451]
[243.0, 494]
```

En este caso, además de la información que mostramos en los demás experimentos, se muestra junto al índice de la instancia recuperada, el número de ocurrencias de palabras en común con la instancia empleada como query.



Como podemos apreciar, tanto en valoración respecto al tiempo, como a las instancias recuperadas, podemos decir que esta técnica mejora con creces las anteriores, ya que, para el mismo número de imágenes, tan solo se ha tardado 1.78 segundos en la construcción del índice invertido, y menos de 0.009 segundos en realizar la consulta. Además, podemos comprobar como las instancias recuperadas presentan una mejor valoración con respecto a las técnicas anteriores, devolviéndonos así instancias claramente visibles.

6.- Valoración

Tras el conjunto de experimentos realizados con diferentes técnicas y métodos de implementación, variando varios parámetros como el número de palabras visuales empleadas en la recuperación de instancias, o el número de imágenes contra el que contrastar la recuperación, hemos visto como la técnica de bolsa de palabras con índice invertido es que mejor resultados nos ha dado, siendo una técnica eficaz y eficiente, que pese a enfrentarse a un mayor número de imágenes, los tiempos obtenidos mejoran notablemente a los de los experimentos anteriores.

Ello se debe ya que, en vez de necesitar hallar la similitud de la instancia a recuperar con cada una de las imágenes posibles, con este modelo directamente accedemos a aquellas imágenes que al menos van a tener una palabra en común con dicha instancia, pues se encuentran dentro del índice de una de las palabras de ésta, lo que nos ahorra cálculos innecesarios con instancias que no comparten nada con la que se pretende recuperar. Se contabilizan así las ocurrencias de instancias dentro de cada palabra y devolvemos las instancias ordenadas en orden decreciente de ocurrencias. La primera de ellas se deberá corresponder consigo misma, y el valor de ocurrencias a su vez, corresponderse con el total de palabras asociadas a dicha instancia.

7.- Referencias

[1] HTTP://WWW.ROBOTS.OX.AC.UK/~VGG/DATA/OXBUILDINGS/OXBUILD_IMAGES.TGZ

[2] HTTP://WWW.ROBOTS.OX.AC.UK/~VGG/DATA/OXBUILDINGS/

[3] HTTPS://DOCS.OPENCV.ORG/3.0-BETA/DOC/PY_TUTORIALS/PY_TUTORIALS.HTML

[4] HTTP://WWW.ALVAROHURTADO.ES/LEER-CARPETAS-Y-ARCHIVOS-CON-PYTHON/

[5] HTTPS://WWW.MICROSOFT.COM/EN-US/RESEARCH/WP-CONTENT/UPLOADS/2007/04/TR-2007-53.PDF

[6] HTTP://WWW.ROBOTS.OX.AC.UK/~VGG/PRACTICALS/INSTANCE-RECOGNITION/INDEX.HTML

[7] MATERIAL DE TEORÍA DE LA ASIGNATURA: RECONOCIMIENTO DE OBJETOS-1: INSTANCIAS

[8] HTTPS://OPENCV-PYTHON-TUTROALS.READTHEDOCS.IO/EN/LATEST/PY_TUTORIALS/PY_ML/PY_KMEANS/PY_KMEANS_OPENCV/PY_KMEANS_OPENCV.HTML#KMEANS-OPENCV

[9] HTTPS://WWW.FLICKR.COM/

[10] HTTP://WWW.ROBOTS.OX.AC.UK/~VGG/DATA/OXBUILDINGS/WORD_OXC1_HESAFF_SIFT_16M_1M.TGZ

[11] HTTPS://JULIOECHEVERRI.WORDPRESS.COM/2015/12/25/MEDIR-TIEMPO-DE-EJECUCION-EN-PYTHON/